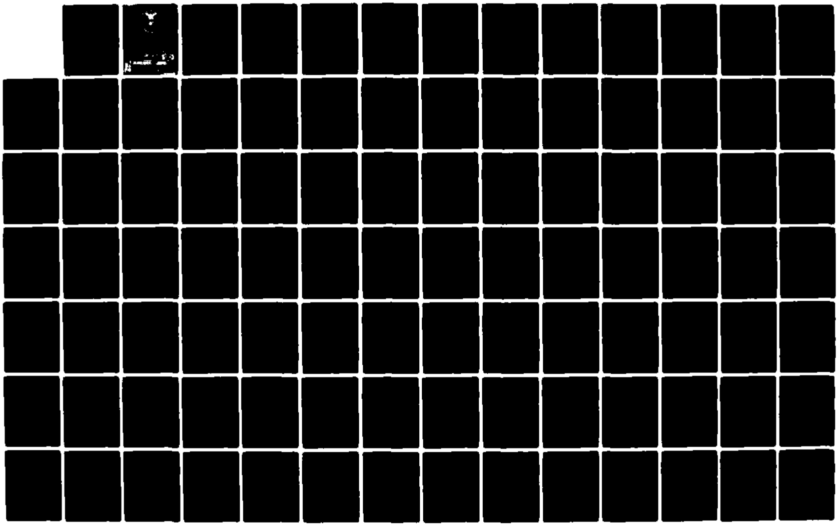
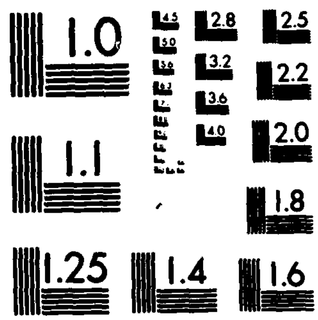
AD-A124 707 DEVELOPMENT OF AN INTERACTIVE CONTROL ENGINEERING COMPUTER ANALYSIS PACKA..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 1/2

UNCLASSIFIED C J GEMBAROWSKI DEC 82 F/G 9/2 NL




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

AD A124707

DEVELOPMENT OF AN
INTERACTIVE CONTROL ENGINEERING
COMPUTER ANALYSIS PACKAGE (ICECAP)
FOR DISCRETE AND CONTINUOUS SYSTEMS

THESIS (VOLUME I OF II)

AFIT/GE/EE/82D-34 Charles J Gembarowski
Major USAF

AFIT/GE/EE/82D-34

DEVELOPMENT OF AN
INTERACTIVE CONTROL ENGINEERING
COMPUTER ANALYSIS PACKAGE (ICECAP)
FOR DISCRETE AND CONTINUOUS SYSTEMS

THESIS (VOLUME I OF II)

Presented to the Faculty of the School of Engineering
of the Air force Institute of Technology
Air Training Command
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Charles J. Gembarowski, B.S.
Major USAF
Graduate Electrical Engineering

December 1982

Preface

This investigation continues the development of the Interactive Control Engineering Computer Analysis Package (ICECAP) begun by Captain Glen T. Logan in his Master's Thesis. This package applies to the design and analysis of discrete and continuous control systems. A VAX 11/780 is used as the host computer for ICECAP.

I wish to thank the AFIT faculty members and students who tested the computer program for me and provided valuable comments. I wish to thank my fellow thesis students Captain Kevin W. Rose, Captain Eugene C. Gilpin, Jr. and Second Lieutenant Steven M. Hadfield for their advice on the use of the VAX 11/780 and for their camaraderie.

I wish to thank my predecessor, Captain Glen T. Logan, for the work he did in starting the ICECAP project and in particular for the assistance and training he provided to help me start my thesis project.

I wish to express my sincere gratitude to my thesis advisor, Dr. Gary B. Lamont, for his invaluable direction and encouragement thoughout the entire thesis endeavor. I also thank the other members of my thesis committee, Dr. Robert E. Fontana, Dr. Peter S. Maybeck, and Mr. John Smith for their efforts in reviewing my thesis and critiquing the program.

Finally, I would like to sincerely thank my devoted and loving wife, Missy, and my two children, Charles and Christopher, for their help, sacrifices and endurance thoughout the entire AFIT assignment.

Table of Contents

VOLUME I AND VOLUME II

Table of Contents

Table of Contents

Table of Contents

APPENDIX J TESTING DOCUMENTATION

List of Figures

Abstract

This thesis reports on an effort to design and implement a modern interactive computer-aided design and analysis package for control systems. This package applies to discrete and continuous time systems. The thesis effort continues the effort begun by Captain Glen T. Logan who used a control engineering design and analysis computer program called TOTAL as his starting point.

This thesis project uses top-down structured analysis and programming techniques to define the new program called ICECAP (Interactive Control Engineering Computer Analysis Package). A user-oriented command lanaguage forms the basic structure of ICECAP. On-line assistance is provided to the user. The program makes use of CRT (Cathode Ray Tube) terminals with a limited graphics capability to improve the user environment.

The program structure allows features to be added in a modular fashion so that others can continue the effort. Emphasis was placed on implementing the continuous time functions first.

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Control system design is a very complicated engineering process. There are many steps involved and sophisticated design tools are required for each step. Furthermore, the control system design process is a cyclic one requiring many iterations that can be very tedious if accomplished manually. Tools to aid in the control systems design process have evolved somewhat independently so that the need for a systems approach to computer-aided design of control systems is apparent.

Two students of the Air Force Institute of Technology (AFIT), Frederick L. O'Brien [52] and Stanley J. Larimer [38] used a systems approach in designing and implementing such a tool. Their system, a very powerful software package, is in use today and is known as TOTAL. TOTAL is hosted on the CDC (Control Data Corporation) Cyber, one of the most heavily used computers at Wright-Patterson Air

Force Base.

Another AFIT student, Glen Logan [41], successfully transported this design tool from the Cyber to the VAX 11/780 in an effort to improve the running environment of TOTAL and to reduce the workload on the Cyber. The computer program in this new environment is known as VAXTOTAL. VAXTOTAL is the foundation upon which this thesis investigation is built. VAXTOTAL is a development version only--it is not meant for release. This thesis effort continues Logan's work. The computer program that results from this thesis investigation will be known as ICECAP (Interactive Control Engineering Computer Analysis Package). This program will be used for the analysis and synthesis of both digital and continuous systems.

1.2 INVESTIGATION OF OTHER TOOLS

There are other control system design and analysis tools available. Appendix D is a synopsis of some of these tools. This appendix includes tools investigated by Logan in his thesis [41] as well as tools found during this thesis investigation. References are given for each tool that is summarized in Appendix D.

TOTAL and VAXTOTAL were chosen as the starting point for this thesis effort because they are Air Force owned, are well established at AFIT, and have proven themselves to be very useful and powerful tools with potential for expansion. The time and budget allotted to a thesis effort does not allow for a costly and lengthy procurement of other tools nor for acquiring the rights to modify commercially available tools.

1.3 PROBLEM STATEMENT

Although it is a very powerful design tool, TOTAL has many deficiencies in its structure and in its usability. Reference Appendix D for a description of TOTAL's capabilities.

1.3.1 Structure - As modern control theory evolves, it will be necessary to extend TOTAL to incorporate new features. TOTAL's structure makes this growth very difficult. The fact that TOTAL had to be segmented into an overlay structure in order to fit it onto the Cyber makes software maintenance more difficult. The use of FORTRAN Common variables makes the modules too tightly coupled.

1.3.2 Usability - TOTAL now favors the experienced user. For example, it requires the user to supply codes in the form of option numbers. Some of the command words that TOTAL uses are not action words, so they cannot be associated with the action that is to be taken. These characteristics make it difficult to learn how to use the system. As a result, new users are discouraged. The problem is especially severe at AFIT where the primary user is the control systems student who does not yet have a firm grasp of control theory. This type of user has the double problem of having to learn new material and the use of a computer-aided design tool. The environment in which the student must use TOTAL (i.e., the Cyber accessed via teletype terminals) is a source of frustration rather than an aid to the learning process.

The main objective of this thesis investigation is to overcome these problems by making the system easy to learn and usable for both experienced and inexperienced personnel and by making the computer environment more pleasant.

1.4 SCOPE

This thesis investigation will design a modern control system computer-aided design tool called ICECAP.

1.4.1 Sound software engineering [37, 64, 66] and human engineering [47, 56, 57] techniques will be used throughout the entire effort.

1.4.2 The thesis effort will revise and incorporate the existing modules from TOTAL that are needed for ICECAP.

1.4.3 The primary structure will be provided through the use of the Pascal programming language [22, 23, 24, 25, 34, 37, 54, 60]. The heart of the computation will be done by the FORTRAN modules if they are best suited for this task. However, the design will be such that either Pascal or FORTRAN can be used for the lower level modules.

This thesis effort will design ICECAP in a highly structured and modular fashion so that subsequent thesis students can understand the current version of ICECAP, complete the system, and maintain the system in an orderly fashion.

1.5 APPROACH

A two-phase approach is used in this development:

1.5.1 Requirements Definition Phase - The major effort in this phase will be to evaluate the existing versions of TOTAL and VAXTOTAL. The major steps to be taken in this phase are as follows:

1.5.1.1 Establish and study the baseline of VAXTOTAL.

1.5.1.2 Run test cases in order to identify errors with TOTAL and VAXTOTAL. Establish a test plan that will be used to determine whether or not VAXTOTAL produces results equivalent to TOTAL.

1.5.1.3 Establish the requirements for ICECAP using past efforts as the point of departure.

1.5.1.4 Demonstrate the feasibility of interfacing a FORTRAN program with a Pascal program on the VAX. This interface is at the heart of Logan's design for ICECAP but was not used in VAXTOTAL.

1.5.1.5 Demonstrate the feasibility, usefulness and acceptance of an on-line capability that will provide assistance when the user demands it and when the program senses that the user needs it.

1.5.2 Design/Implementation Phase - This phase consists of performing and documenting the following tasks:

1.5.2.1 Design a program structure that will allow follow-on thesis students to design, test, and modify the computer program in stages.

1.5.2.2 Correct errors in the present implementations of TOTAL and VAXTOTAL including correcting the major inconveniences in using the program.

1.5.2.3 Implement and interface the computer program modules as time permits.

1.6 OVERVIEW OF THESIS

This thesis is contained in two volumes. Volume I contains all of the chapters and Appendices A through F. Volume II contains Appendices G through J. The major chapters and appendices are briefly summarized as follows:

Chapter 2 provides the requirements definition for ICECAP in the form of a specification with an emphasis on the functional requirements. The specification is meant to cover the requirements for the entire ICECAP project. This thesis implements only a subset of this specification.

Chapter 3 documents the design structure for ICECAP. This chapter is designed to transfer the "corporate memory" of "why the design is what it is" to follow-on thesis students.

Chapter 4 documents the implementation of ICECAP. Voluminous details of the design are provided in appendices and separate documents as appropriate. These details are provided so that other thesis students can continue the effort.

Chapter 5 defines the requirements on how to test ICECAP. These tests will establish whether or not ICECAP satisfies the requirements defined in Chapter 2.

Chapter 6 contains the conclusions reached in this thesis investigation and gives recommendations regarding follow-on efforts.

Appendix A contains the structure charts for ICECAP in alphabetical order. These charts establish the program modules needed for ICECAP and show the data and the control flow through the program.

Appendix B is the data dictionary for the ICECAP modules. The dictionary entries are in alphabetical order and give an overview of the function of each module and the parameters and variables used in each module.

Appendix C is a set of reports that document problems with TOTAL, VAXTOTAL, and ICECAP. These reports serve as a basis for correcting the deficiencies in these three programs.

Appendix D is a synopsis of existing and planned computer-aided design packages for control systems.

Appendix E contains the definition of the ICECAP command language in flow chart form. These charts provide an unambiguous definition of the language.

Appendix F contains descriptions of the new and revised FORTRAN modules used in ICECAP.

INTRODUCTION

Appendix G contains the Pascal source code for ICECAP.

Appendix H contains FORTRAN source code for ICECAP. The modules are listed in alphabetical order. Only new and revised modules are included. The unchanged modules from TOTAL and VAXTOTAL have been documented by Logan in the form of unpublished source code listings.

Appendix I contains information about where the important computer files relating to this thesis effort are located.

Appendix J contains the relevant documentation resulting from testing ICECAP.

These chapters and appendices are meant to be "living" documents. They are expected to change as the result of follow-on thesis efforts as more and more of ICECAP becomes defined and implemented.

CHAPTER 2

REQUIREMENTS DEFINITION

2.1 INTRODUCTION

This chapter forms the systems requirements specification for the entire ICECAP project. The meanings of requirements priority categories are first defined. Then the requirements are stated in logical order, with the priority defined for each requirement. Functional requirements, human engineering requirements, and software engineering requirements are included. This thesis effort will implement a subset of these requirements, viz., the priority one requirements. It is anticipated that follow-on thesis efforts will implement the other requirements. Finally, the testing requirements are stated and organized into four major categories of functional requirements testing, program flow testing, on-line assistance capability testing, and output capability testing.

2.2 DEFINITIONS OF PRIORITIES

The requirements below have been categorized into one or more of three priorities in order to facilitate dividing out the entire ICECAP project into meaningful thesis efforts. The definitions of these priorities are as follows:

2.2.1 Priority One - This priority means that this requirement must be at least partially contained in the initial program design in order to have a running program with which to demonstrate both feasibility and capability. This may involve both requirements that are already satisfied by VAXTOTAL and requirements that are new to ICECAP. This category generally refers to all of the human interface requirements, to the most important features required for continuous time design and analysis, and to the software engineering requirements. Continuous time features are considered a higher priority than the discrete time features because generally students are taught the fundamentals of continuous control systems before the fundamentals of discrete control systems.

2.2.2 Priority Two - This priority refers to requirements that are at least partially implemented in VAXTOTAL but need not be implemented in ICECAP at this time in order to demonstrate feasibility and capability. This category generally refers to the matrix manipulation features, the discrete time design and analysis features and the remainder of the continuous time design and analysis features.

2.2.3 Priority Three - This category refers to the functional requirements needed to have a complete control system computer-aided design package. It includes functional requirements related to the control system design area presently within the state of the art but not yet implemented in VAXTOTAL. Examples of these kinds of requirements are the stochastic estimation and control requirements.

2.3 REQUIREMENTS

ICECAP shall assist the user in performing conventional, modern, and stochastic control system design and analysis for both discrete and continuous systems. The specific requirements and their priorities (in parentheses) are as follows:

2.3.1 Functional Requirements - ICECAP shall provide the following functional capabilities:

2.3.1.1 Transfer Function Manipulation (1)

2.3.1.1.1 Open Loop Transfer Function (1)

2.3.1.1.2 Closed Loop Transfer Function (1)

2.3.1.1.3 Forward Transfer Function (1)

2.3.1.1.4 Feedback Transfer Function (1)

2.3.1.1.5 Return Difference Transfer Function (2)

2.3.1.1.6 Block Diagram Manipulation (2)

2.3.1.2 Matrix Algebra (2)

2.3.1.2.1 Matrix Addition (2)

2.3.1.2.2 Matrix Subtraction (2)

2.3.1.2.3 Matrix Multiplication (2)

2.3.1.2.4 Matrix Inversion (2)

2.3.1.2.5 Matrix Transposition (2)

2.3.1.2.6 Matrix Factorization (2)

2.3.1.2.7 Matrix Square Roots (2)

2.3.1.2.8 Solve Ax=b (2)

2.3.1.2.9 Singular Value Decomposition (2)

2.3.1.3 Polynominal Manipulation (2)

2.3.1.4 Calculator Functions (2)

2.3.1.5 State Variable Equation Manipulation (2)

2.3.1.6 Control System Definition (1, 2)

2.3.1.7 Steady State Response Analysis (1)

2.3.1.8 Transient Response Analysis (1)

2.3.1.9 State Transition Matrix Evaluation (2)

2.3.1.10 State Equation Solver (2)

2.3.1.11 Laplace Transformations (2)

2.3.1.12 Inverse Laplace Transformations (2)

2.3.1.13 Partial Fraction Expansion (1)

2.3.1.14 Frequency Response Evaluation (2)

2.3.1.14.1 Bode Plots (2)

2.3.1.14.2 Direct Polar Plots (2)

2.3.1.14.3 Inverse Polar Plots (2)

2.3.1.15 Stability Analysis (2)

2.3.1.15.1 Routhian Array (2)

2.3.1.15.2 Nyquist (2)

2.3.1.15.3 Nichols Plots (2)

2.3.1.16 Steady State Error Analysis (2)

2.3.1.17 Root Locus Analysis (1)

2.3.1.18 Root Locus Compensation (2)

2.3.1.18.1 Cascade Compensation (2)

2.3.1.18.2 Feedback Compensation (2)

2.3.1.19 Closed Loop Pole-Zero Assignment (2)

2.3.1.19.1 Guillemin-Truxal Design (2)

2.3.1.19.2 State Variable Feedback (2)

2.3.1.20 Algebraic Riccati Equation Solver (3)

2.3.1.21 Z Transformation (2)

2.3.1.22 Inverse Z Transformation (2)

2.3.1.23 Digital Computer Compensation (2)

2.3.1.24 Filter Design (2)

2.3.1.25 Stochastic Estimation and Control Design and Analysis (3)

2.3.1.25.1 Kalman Filter Design for Continuous and Discrete Time Measurements (3)

2.3.1.25.2 Analysis of Kalman Filter Design (3)

2.3.1.25.3 Square Root Filtering (3)

2.3.1.25.4 U-D Covariance Factorization Filtering (3)

2.3.1.25.5 Weiner Filtering (3)

2.3.1.25.6 Optimal Smoothing (3)

2.3.1.25.7 LQG (Linear Quadratic Gaussian) Controller Design for Continuous and Discrete Time Systems (3)

2.3.1.25.8 Observer and Full-State Feedback Controller Design via Pole-Placement Methods (3)

2.3.1.26 Z-Domain Stability Analysis (2)

2.3.1.26.1 Jury-Blanchard Test (2)

2.3.1.27 S-Domain to W-Domain Transformations (2)

2.3.1.28 S-Domain to Z-Domain Transformations (2)

2.3.1.28.1 First Backward Difference (2)

2.3.1.28.2 Tustin Transformation (2)

2.3.1.29 Pseudo Continuous Time Control System Analysis and Synthesis (2)

2.3.1.30 Compensator Design (2)

2.3.1.30.1 Direct (DIR) Method (i.e, all design is done in the Z-domain) (2)

2.3.1.30.2 Digital (DIG) Method (i.e., the design is started in the S-domain and then transferred to the Z-domain) (2)

2.3.2 Human Engineering Requirements - ICECAP shall be user friendly:

2.3.2.1 ICECAP shall provide on-line assistance in its use upon demand. (1, 2, 3)

2.3.2.2 ICECAP shall be command oriented. A means to assist the users in formulating commands shall be provided. This assistance shall not be distractive. It shall not impede those users who do not need on-line assistance. (1)

2.3.2.3 ICECAP shall provide instruction in the various aspects of control theory through some sort of teaching facility. (2, 3)

2.3.2.4 ICECAP shall notify the users when they have erred in providing input. This shall be done in a non-hostile manner using meaningful error messages. (1)

2.3.2.5 ICECAP shall provide a facility for providing meaningful and selective printed output as the means of documenting the users' designs of control systems. (1)

2.3.2.6 ICECAP shall provide a means of storing the essentials of a design in progress so that the users may continue their designs at other sessions. (1)

2.3.2.7 ICECAP shall provide a capability for the users to define command strings so that they may iterate a design without having to type in the same commands repeatedly. This shall include a facility for the users to specify data as part of the command string. (2)

2.3.3 Software Engineering Requirements - ICECAP shall be designed using sound software engineering principles such as those advocated in the software engineering literature [37, 64, 66] so that it can be easily maintained and augmented. (1)

2.3.3.1 ICECAP shall be as portable as reasonable, i.e., it shall be capable of being rehosted on other VAX's. (1)

2.3.3.2 ICECAP shall be modular. (1)

2.3.3.3 ICECAP shall use loosely coupled modules as much as possible. (1)

2.4 TESTING REQUIREMENTS

ICECAP shall be tested as follows:

2.4.1 Functional Requirements - The functional requirements that have been implemented in ICECAP shall be tested using known test cases, such as the sample problems in the student handouts [30]. It shall be determined whether or not ICECAP provides results consistent with (or better than) the results of using TOTAL for these problems. ICECAP numerical results shall be considered consistent when they are the same as the results of TOTAL within a tolerance of 0.001. ICECAP results shall be considered better than TOTAL's results when a deficiency in TOTAL has been corrected in ICECAP.

2.4.2 Program Flow - ICECAP shall be run to determine whether or not the program flows properly. The program must transistion to valid known states. The program must not hang in an endless loop. The ability to exit gracefully from the program must be demonstrated.

2.4.3 On-line Assistance - The ability of the intended user to formulate the commands necessary to design and analyze a control system with the on-line assistance that is provided must be demonstrated.

2.4.4 Output Capability - The ability to document a control system design and analysis selectively and conveniently must be demonstrated.

2.5 SUMMARY

The systems requirements specification for the entire ICECAP project has just been presented. Priority categories have been established and their meanings defined. The functional requirements, human engineering requirements, and the software engineering requirements were stated and priorities were assigned to each requirement. Finally, the testing requirements were stated and organized into four major categories of functional requirements testing, program flow testing, on-line assistance capability testing, and output capability testing.

CHAPTER 3

DESIGN STRUCTURE

3.1 INTRODUCTION

This chapter addresses the design structure for those requirements stated in Chapter 2 that are to be implemented as a result of this thesis effort. Basically, the chapter addresses the priority one requirements but at the same time dictates a design structure that will support the implementation of all of the design requirements at some future time.

The chapter begins with an outline of the top-level requirements that affect the ICECAP design structure. Next, the constraints that influence the structure are addressed. The approach used to derive the design structure is provided. Finally, how the need for user "friendliness" and how the computer environment are related to the structuring of the ICECAP design are addressed.

3.2 REQUIREMENTS SUMMARY

This section outlines the top-level requirements that drive the design structure of ICECAP. The paragraph references to the requirements of Chapter 2 are provided.

3.2.1 Specifically, ICECAP shall be designed to assist the user in performing control system design and analysis for both discrete and continuous systems (cf. para. 2.3).

3.2.2 Certain basic functional capabilities shall be provided in the initial design of ICECAP in order to have a working system and to prove the concept so that further implementation of functional capabilities can proceed. These initial capabilities shall include: transfer function manipulation (cf. para. 2.3.1.1); steady state response analysis (cf. para. 2.3.1.7); and root locus analysis (cf. para. 2.3.1.17).

3.2.3 ICECAP shall be designed to be user "friendly". Specifically, ICECAP shall provide on-line user assistance in the form of prompting upon demand; ICECAP shall allow the user to specify action by explicit commands that are readily recalled by the user rather than by cryptic codes which must be looked up in a manual; ICECAP shall provide error checking and notification in a clear and non-hostile manner (cf. para. 2.3.2).

3.3 DESIGN CONSTRAINTS

The design of ICECAP is constrained by several factors including the size of VAXTOTAL, the availability of resources, and the amount of time available for the thesis project.

3.3.1 Program Size - Because of its large size, a complete redesign and rewrite of VAXTOTAL is not practical. This constraint dictates that ICECAP be the combination of a new program written in a language suitable for processing user commands and of selected modules from VAXTOTAL written in FORTRAN. The FORTRAN modules should be capable of performing the powerful calculations that a control system design tool demands.

3.3.2 Availability Of Resources - The ICECAP design is also constrained by the existing computer facilities and computer programming languages available at AFIT. This includes the existing input and output devices presently available at AFIT. These constraints have very positive aspects since they drive the use of a virtual memory system, the use of interactive terminals, and the use of the Pascal language. The input and output devices constraint means that there can be very little plotting capability designed into the program at this time. However, the design must allow for a plotting capability to be added at a later time.

3.3.3 Time - The amount of time that any one thesis student can be reasonably expected to devote to such a large project definitely constrains the design. This constraint dictates that the program be designed in a modular fashion using a tree structure so that new features can be implemented later. This constraint also dictates that the program be well documented.

3.4 DESIGN APPROACH

The design approach for ICECAP was to use top-down structured analysis. This approach developed into five major steps as follows:

3.4.1 Develop The Gross Data Flow Diagram - The first step was to develop the gross data flow diagram shown below.

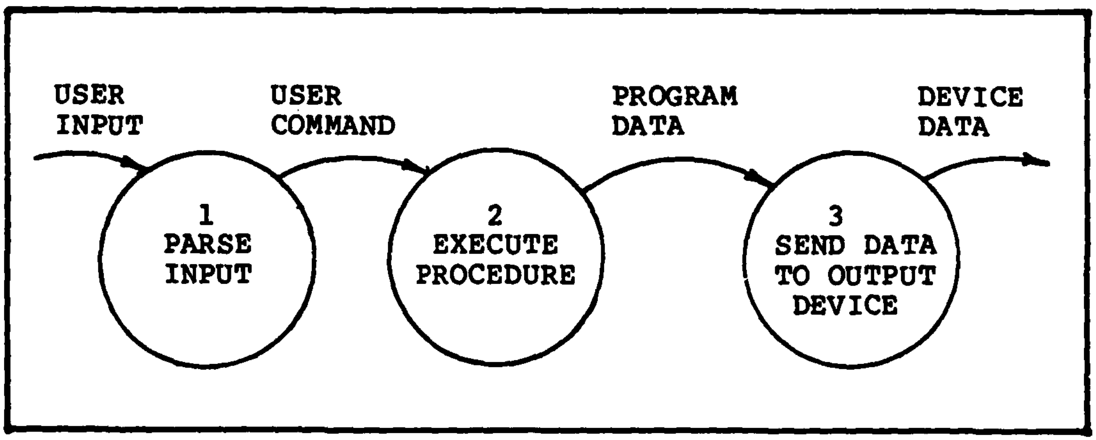


Figure 3-1. ICECAP Gross Data Flow Diagram

The above diagram was developed by Logan [41: 48].

Basically, the user provides input which is then interpreted as a command. The command then executes a procedure or a series of procedures at which time data may be required of the user. The execution of the procedure(s) results in data going to an output device or file.

3.4.2 Develop The Overall Program Structure - Next, the overall program structure was developed as shown below.

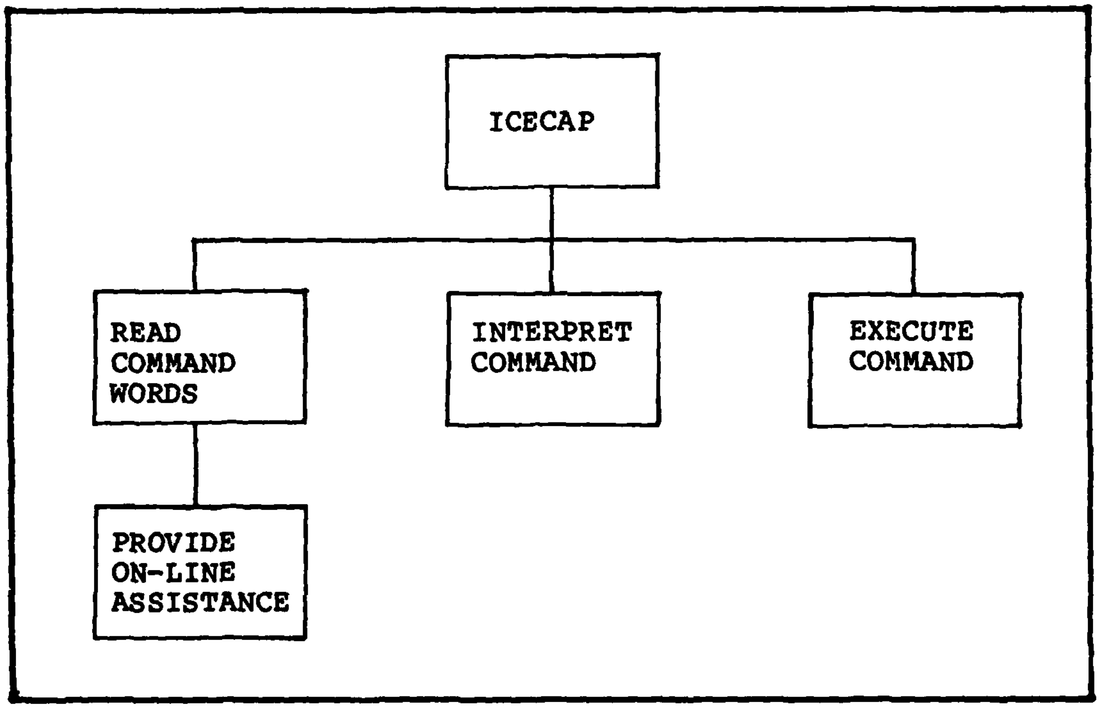


Figure 3-2. ICECAP Overall Structure Chart

Commands or partial commands input by the user are read and on-line assistance is provided as necessary to aid the user in formulating complete commands. Once a command is fully formed, the interpret modules decode the command. Once decoded, the commands are translated. The translation is then passed to a set of modules that actually execute the command. The process repeats until the user stops the program.

3.4.3 Define Syntax For One Command - Next the syntax diagram was developed for one specific command, the DEFINE command. This command is of sufficient complexity so as to ring out the ability of the design structure to support the processing of multi-word commands. The details of the syntax for this command are found in Appendix E.

3.4.4 Develop The Modules - The modules to implement the overall design structure and to implement the DEFINE command specifically were then developed. The details of this implementation are the subject of Chapter 4.

3.4.5 Iterate The Process - With the overall structure now developed and one command implemented and working, the development process now repeats and the syntax diagrams for other commands are developed and implemented one by one. The program thus grows in a tree-like fashion. Also, the structure and the modules are refined in this step.

3.5 USER "FRIENDLINESS"

The user is the reason why ICECAP exists. Special attention has been given to making the system "friendly" to the user. Special prompting features have been built in to the system so that a new user can quickly learn and enjoy using the system. These features have been designed in such a way that an experienced user is not distracted by the on-line help that is available.

The system is able to sense when the user needs prompting and provides only that prompting which is needed. The design structure for this process is as follows. The user issues a carriage return as a signal for the system to respond. There are three cases of how the system responds.

3.5.1 Invalid Command - If the user has entered an invalid command, the system responds with a message as to the nature of the error and then user has the opportunity to re-enter the command.

3.5.2 Incomplete Command - If the user has entered an incomplete command, the system responds with choices for the next command word. The design structure allows a short explanation of the nature of each of the choices. The user simply types in one of these choices.

3.5.2.1 Complete And Valid Command - If the user has entered a complete and valid command, the system responds by executing the command.

3.6 ENVIRONMENT

The computer environment determines how "user friendly" a system can be made. The design structure of ICECAP capitalizes on the pleasant computer environment that the host computer affords in order to maximize the user "friendliness".

3.6.1 Screen Terminal - The design structure of ICECAP takes advantage of the fact that the user will be using a screen terminal rather than a printing terminal (although the structure can accommodate the use of a printing terminal if one is desired). These advantages are faster response time, the ability to highlight important information, the ability to use graphics, and the ability to review information before printing it.

3.6.1.1 Response Time - Because the system has a fast response time, ICECAP is structured to provide the user with more meaningful information. For example, as far as the user can tell, it takes no longer to print the menu of valid initial commands words and the prompt asking the user for input than it does to print only the prompt. This menu certainly makes the working environment more pleasant for the user at no additional time expense.

3.6.1.2 Highlighting - The structure of ICECAP provides the capabilty to highlight text at anytime that highlighting is needed. This gives the ICECAP programmer the capability to emphasize command words so that the user can learn the ICECAP command language much more quickly. The user remembers the highlighted words and can associate these words as building blocks for valid ICECAP commands. The structure charts of Appendix A show that ICECAP has been designed in such a way that highlighting can be turned on and off by subprogram calls.

3.6.1.3 Graphics - Similarly, ICECAP has been structured to support a capability to turn the terminal graphics on and off by subprogram calls. This can be seen by reading the structure charts in Appendix A. This gives the programmer the opportunity to enhance the user's understanding of control theory and the use of the design tools by drawing control system block diagrams on the terminal.

3.6.1.4 Selective Printing - One of the most aggrevating aspects of doing a computer-aided design on a printing terminal is that the resulting output is a running tally of all of the design iterations and user errors. The result is that it is very difficult to walk away from the design session with a neat bottom line design suitable for submission to an instructor or to an employer. At best, the user sorts through the printout which may be several yards long and tries to recap the important information and re-enter it so that only the end result is printed out. The design structure of ICECAP makes this tedious and error-prone re-entry process totally unnecessary. ICECAP allows the user to review all output on the screen. Once satisfied with the results, the user can than cause the output to be written to a file which can be neatly printed out in as many copies as desired after the design session. The modularity of the ICECAP design structure makes this selective printing possible.

3.6.2 No Need For User's Manual - The fact that the design structure of ICECAP allows on-line assistance to be developed for a command at the same time that the ICECAP command is implemented in the language obviates the need for a user's manual. The user's manual is actually provided on line. This makes the environment more pleasant for the user because there usually is very little room at the terminal station to accommodate user's manuals, especially when the user needs to have textbooks and worksheets there for the specific design problem at hand.

3.6.3 Files Storage - One of the most important features needed for a pleasant user environment is the ability to store files so that a control system design can be continued at another session. To re-enter all previous data in order to continue a design session is a very tedious and error-prone task. ICECAP has been structured to take advantage of the files management system of the VAX 11/780 and to allow the user to store current designs and several previous designs so that any design session may be continued at any time.

3.7 SUMMARY

This chapter has addressed the design requirements and constraints that have influenced the ICECAP design structure. In particular, the role of the computer environment and the requirement for a "friendly" user interface in establishing that structure has been addressed. The major steps taken in applying top-down structured analysis to derive the design structure were described.

CHAPTER 4

IMPLEMENTATION

4.1 INTRODUCTION

This chapter describes the details of the overall architecture of ICECAP. This description is from the viewpoint of how the program flows in order to process and execute any user initiated ICECAP command. The details of how the program flows for all possible ICECAP commands are too voluminous and cannot be treated in this chapter. That amount of detail is left to the appendices. However, examples of a specific command are given in order to show the architecture for the general case.

4.2 SYSTEM DESIGN

ICECAP is designed using top-down structured programming techniques [37, 64, 66]. A gross flow chart is provided in Figure 4-1. The major activities are as follows:

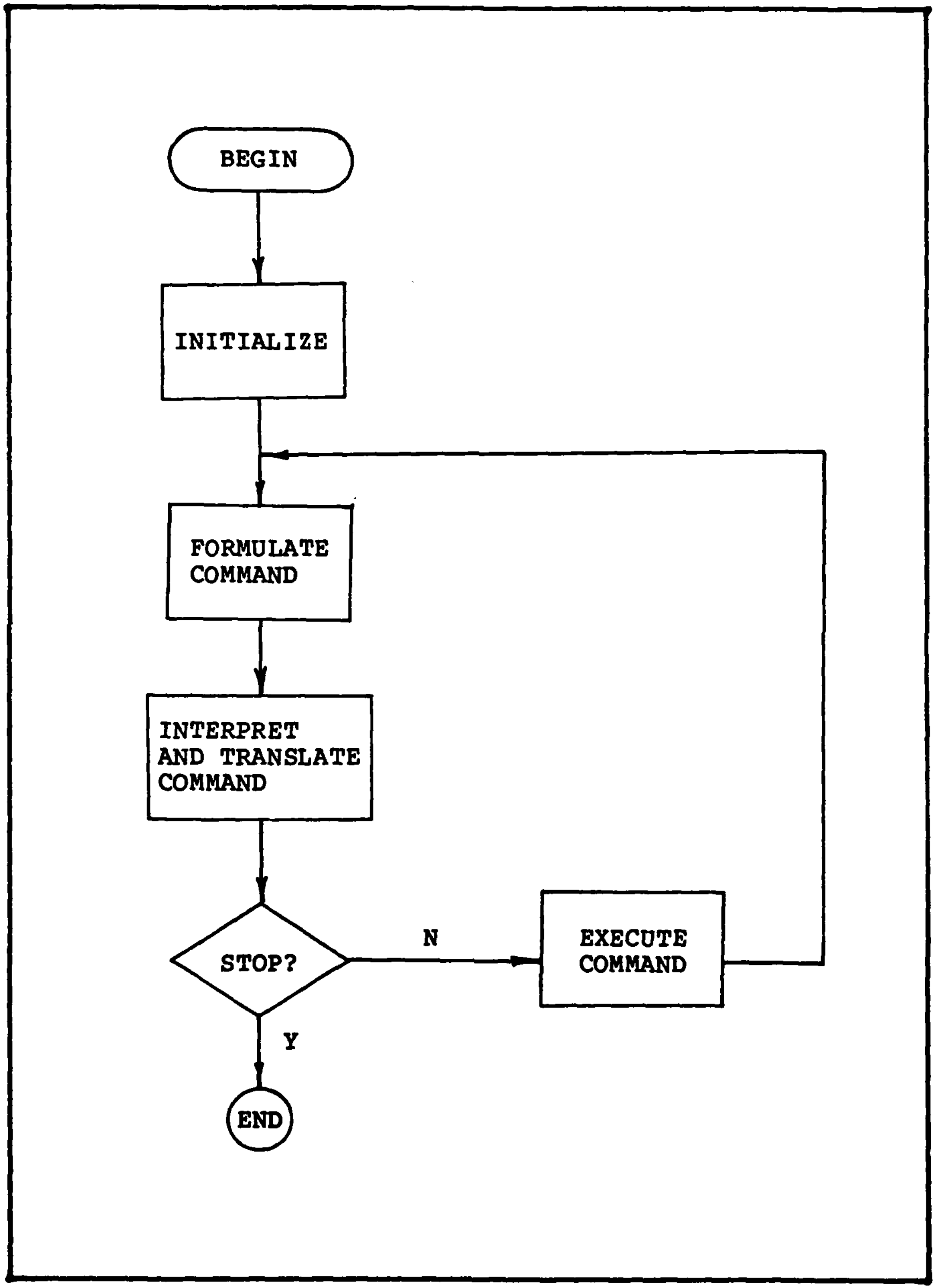


Figure 4-1. ICECAP Gross Flow Chart

4.2.1 Program Initialization - This activity includes the normal things that are done to initialize any program, i.e., clear the screen, display the initial title slide, set certain values to zero and set variables and flags to their initial values.

4.2.2 Command Formulation - This activity begins with a prompt in the form of a menu-like header that shows all of the valid command words that can be used to start a valid ICECAP command string. As words are selected, further prompting is available to indicate the valid choices for the next word in the command string. If the next command word is known, the user is not bothered by the prompting. This process continues until a valid command string is formed.

4.2.3 Command Interpretation And Translation - Once a valid command string is formed its meaning must be interpreted so that it may be translated for use by the FORTRAN modules that carry out the actual command.

4.2.4 Command Execution - Except for changes to improve performance, the execution of the commands is done basically in the same manner as is now done in VAXTOTAL. The details of the various VAXTOTAL commands have been documented in previous theses [38, 41] and user manuals [30, 39] and will not be repeated here.

4.2.5 Return Of Control - Control is then returned to the main program after the command has been executed. A new command can then be entered, interpreted, translated, and executed. This process repeats until the user indicates that the user has finished by entering the command "STOP".

4.3 ALGORITHMS

The algorithms associated with the activities associated above are now described.

4.3.1 Program Initialization - The initialization activity consists of clearing the screen, setting the terminal parameters, displaying an initial screen showing copyright information and program identification information, and initializing values.

4.3.1.1 Clearing The Screen - This process is carried out by the Pascal procedure called CLEAR which is a terminal-unique message that is written to the terminal.

4.3.1.2 Setting The Terminal Parameters - This process is carried out by the main program by sending a terminal-unique message to the terminal to put the screen into the inverse video mode. The inverse video is the mode of dark letters on a light background. The use of inverse video is strictly a matter of taste. It provides a brighter screen which is more pleasing to the eye. The user can override this setting by using the terminal set-up keys.

4.3.1.3 Initial Screen Display - This process is carried out by the Pascal procedure TITLE_SLIDE which, among other things, puts the terminal into the graphics mode, paints the word ICECAP onto the screen, draws a box around the word ICECAP, displays the authors, and shows the copyright information. Effort was put into designing the title slide in this manner in order to emphasize that the user is entering into the program and to give the program a professional appearance. It is important that the users perceive that they are using a professional program, if they are to have confidence in the results.

4.3.1.4 Initialization - Initialization of the Pascal portion of the program is carried out by the main program. Initialization of the FORTRAN portion of the program is carried out by the new FORTRAN module called TOTINI which sets various control flags as well as setting variables to their inital values.

4.3.2 Command Formulation - The command formulation activity consists of prompting the user on valid initial command words, reading the command words, checking the validity of the command words, and prompting the user on valid choices for the next word in the command.

4.3.2.1 Initial Command Word Prompt - The prompting activity is done by the Pascal subprogram HELP_PROMPT. This subprogram prints a header which lists all of the available initial command words highlighted with light letters on a dark background in contrast to the remaining text which is dark letters on a light background. Because of the highlighting the user is able to learn the initial command words more quickly and therefore can eventually disable this prompting feature. Highlighting is done by terminal-unique commands sent to the terminal as a result of calls to subprograms HIGHLIGHT and NOHIGHLIGHT.

4.3.2.2 Reading The Command Words - The command word reading and collecting is done by the Pascal subprogram READCOM. Reference Figure 4-2. READCOM displays the ICECAP prompt (ICECAP>) as a signal for the user to begin entering a command. Unnecessary blanks are trimmed by the Pascal subprogram TRIM and the command word(s) are put in uppercase by the VAX Library Routine named STR$UPCASE. This makes the later parsing of the commands simpler yet allows the user to enter commands in lower, upper, or mixed case. The reading process continues until the user issues a carriage return signalling to the program that the command is complete or that a prompt is needed for the next command word. A flag with the variable name RESOLVED is used to indicate whether or not the command string is a complete

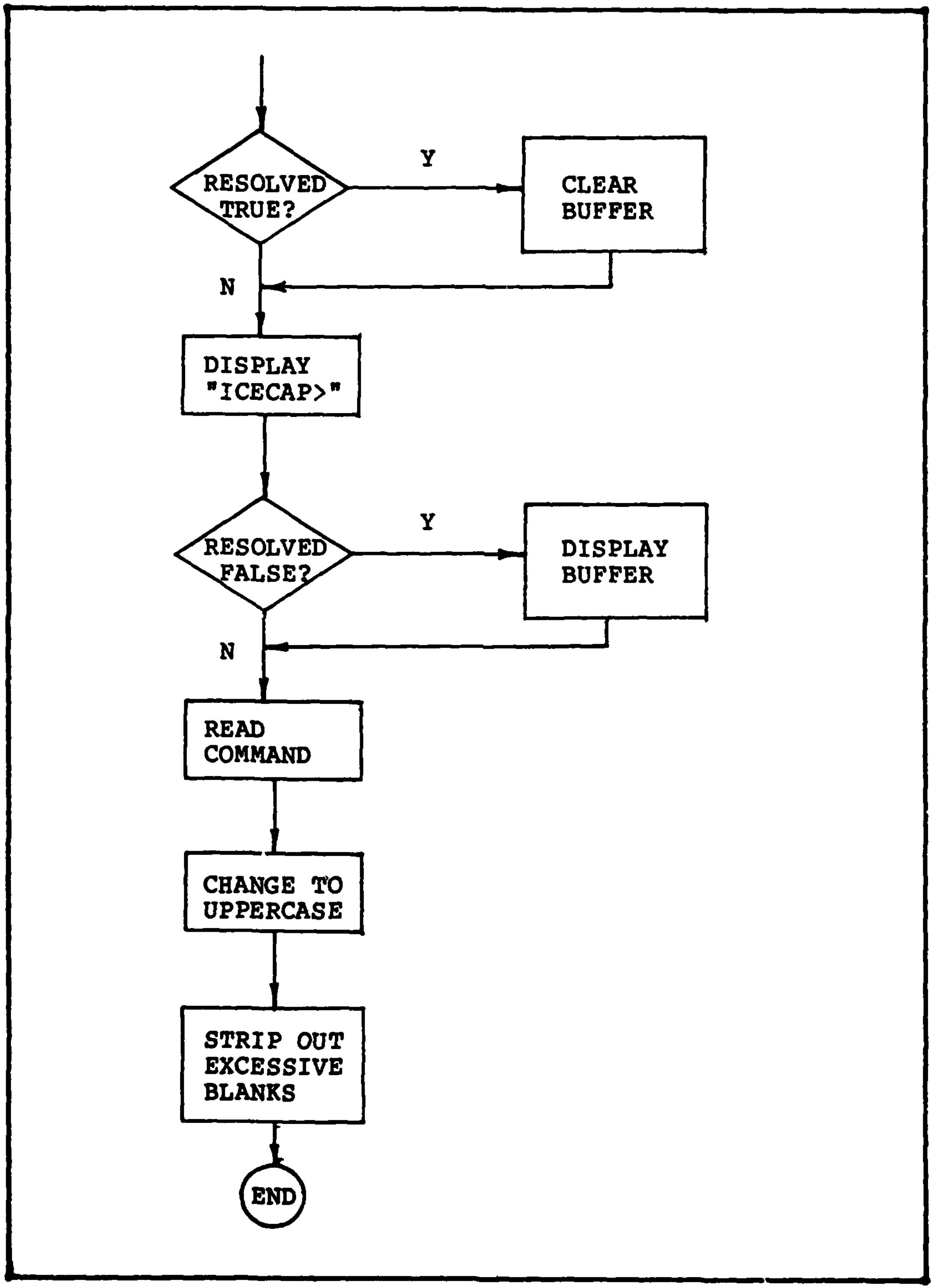


Figure 4-2. Flow Chart for Procedure READCOM

one. If RESOLVED is false, procedure READCOM reads the next command word(s) as a continuation of the previously entered command word(s). If RESOLVED is true, READCOM reads the next command word(s) as a new command.

4.3.2.3 Validity Checking Algorithm - Validity checking is done in two ways as follows:

4.3.2.3.1 First, each command word is checked against the ICECAP dictionary of valid command words. Reference Figure 4-3. This is done by the Pascal subprogram DICTIONARY. If any word used in the command string is not a valid ICECAP command word the user is so notified. Abbreviations are allowed and also appear in the dictionary. If a valid abbreviation is used in a command string, the subprogram DICTIONARY substitutes the expanded version of the word for the abbreviation. For example, if the command word that the user enters is AMA, then procedure DICTIONARY changes that command word to AMAT. This makes the parsing simpler later and teaches the user the full spelling of the ICECAP command words. If all words in the command string are valid command words, the second phase of the validity checking begins.

4.3.2.3.2 The first command word in the command string is checked to see if it is a valid initial command word. This is done by the Pascal subprogram INTERPRET. If the first command word the user has entered is not a valid initial

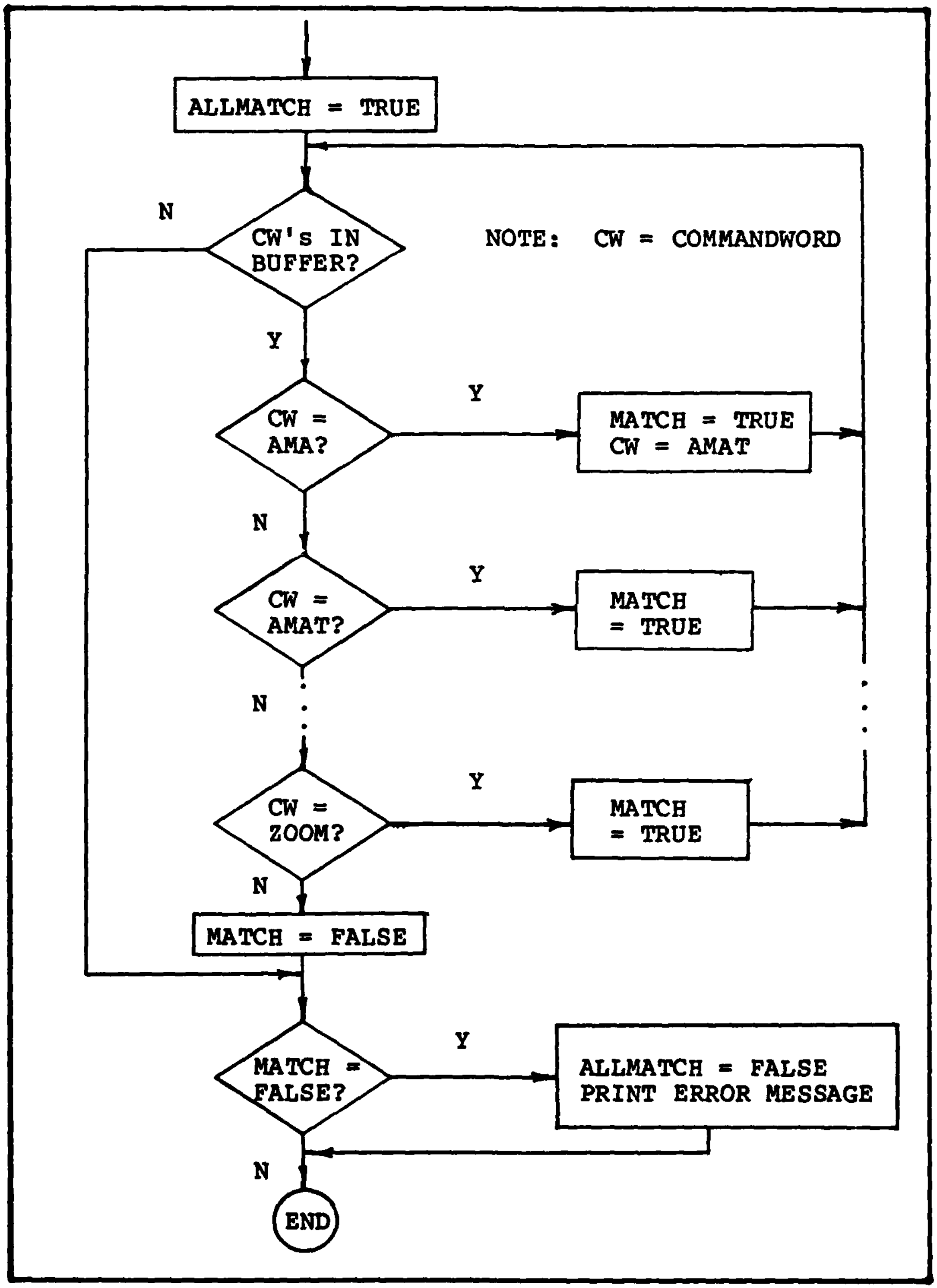


Figure 4-3. Flow Chart for Procedure DICTIONARY

command word the user is so notified and the ICECAP prompt is displayed so that another command may be entered. Depending on what the initial command word is, the subprogram INTERPRET branches out in a tree-like fashion to other Pascal subprograms to look for valid objects of the initial command word and notifies the user if an invalid object has been encountered. This branching out and validity checking continues until the command string is finished and the command is ready for execution.

4.3.2.3.3 Prompting - Prompting is given when the program senses that the user needs information on what the next word in the command string can be. The user signals the need for this information by issuing a carriage return anywhere in the command formulation process. Upon receiving this signal the program displays the valid choices for the next command word. This is done in a tree-like manner wherein each Pascal subprogram displays the choices that are appropriate for the command that has been partially formulated. Reference Figure 4.4. For example, in formulating the command DEFINE GTF POLY, the Pascal subprogram DEFINE calls Pascal subprogram DEFINE_PROMPT in order to prompt the valid choices for the second command word. Once the user has selected the second command word, in this case GTF, then the Pascal subprogram DEFINE calls Pascal subprogram DEFINE_TF which then prompts the user to choose the third command word, in this case POLY.

4.3.2.4 Command Interpretation And Translation - Command Interpretation is made via the program flow. Reference Figure 4-4. The choice of certain command words that make up the command string cause certain If-Then-Else statements to execute. If-Then-Else statements are used to direct the flow because the more convenient case statements do not work for strings in Pascal. (However, for clarity, the decisions in Figure 4-4 are shown with case statements.) The interpretation of the entire command comes from having executed certain combinations of If-Then-Else statements. Once the interpretation is known then the translation is made directly to option numbers that the current version of VAXTOTAL normally receives directly from the user. This translation is passed to the FORTRAN modules as if they were coming directly from the user as coded VAXTOTAL option numbers or VAXTOTAL commands. How this is done is described in the next section.

4.4 PASCAL/FORTRAN INTERFACE

The main Pascal/FORTRAN interface is the FORTRAN module TOTICE (named for TOTAL and ICECAP). The Pascal portion of ICECAP interprets the user commands and translates them either to VAXTOTAL option numbers or to VAXTOTAL commands.

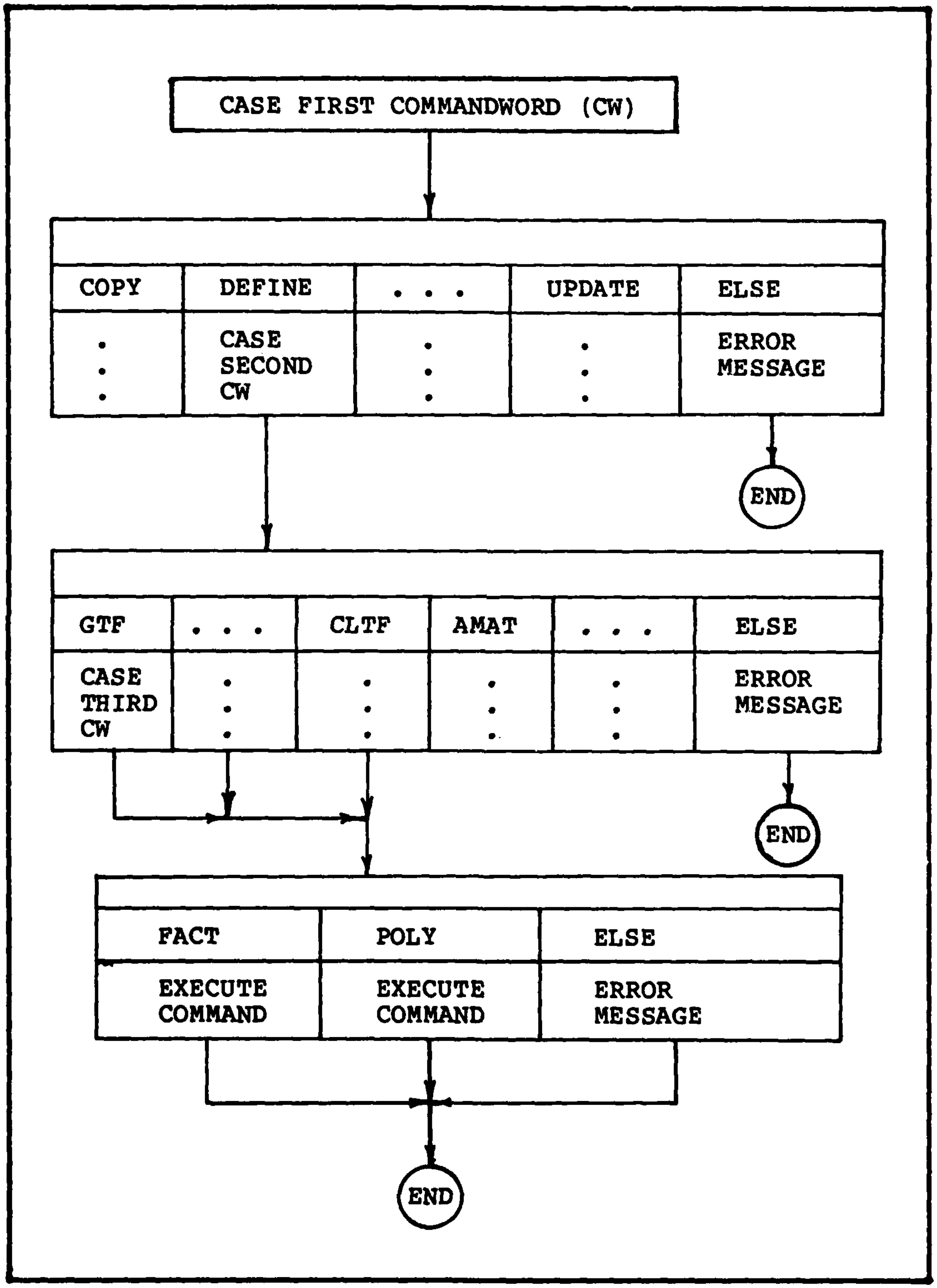


Figure 4-4. Flow Chart for Procedure INTERPRET

4.4.1 Option Numbers - If the command translates to an option number, then the option number is passed to TOTICE via the first element in the argument list (OPTIONNUMBER). If this is the case, the second element in the argument list is ignored. The FORTRAN modules then react to this option number just as if the number had been entered by the user using VAXTOTAL.

4.4.2 Commands - If the command translates to a VAXTOTAL command, then option number zero is passed to TOTICE via the first element in the argument list (OPTIONNUMBER) to signal the fact that the second element in the argument list (LINE) is a command. This command is then passed to the FORTRAN modules to be processed as if the command had been entered by the user using VAXTOTAL.

4.5 DESIGN DOCUMENTATION

The design documentation has been logically arranged in the appendices of this thesis for the convenience of the reader and to facilitate program maintenance. Program maintenance refers to correcting logical errors in the existing program as well as to continuing program development. The appendices which contain this design information are as follows:

- o Appendix A, Structure Charts
- o Appendix B, Data Dictionary

- o Appendix E, Command Language Definition
- o Appendix F, FORTRAN Module Descriptions
- o Appendix G, ICECAP Pascal Source Code
- o Appendix H, ICECAP FORTRAN Source Code

4.6 SUMMARY

The design implementation and rationale for the implementation decisions of ICECAP have just been presented. Details on how the program is initialized and on how commands are formulated, translated and executed have been presented. Details on specific commands are out of scope of this chapter because the heart of the design implementation is in the general command processing capability. This chapter would have been too voluminous if it had treated every possible ICECAP command. The details of each specific command are shown in Appendix A, Structure Charts and Appendix E, Command Language Definition.

CHAPTER 5

TESTING

5.1 INTRODUCTION

This chapter covers the entire program for testing ICECAP. It addresses the testing requirements, the plan for conducting the testing, the results of the testing and all of the documentation generated as a result of the testing.

5.2 TESTING REQUIREMENTS

The requirements for testing ICECAP as stated in Chapter 2 are repeated here for convenience and completeness.

5.2.1 Functional Requirements - The functional requirements that have been implemented in ICECAP shall be tested using known test cases, such as the sample problems in the student handout on the use of TOTAL [30]. This handout contains several printouts of actual interactive sessions. It shall be determined whether or not ICECAP provides results consistent with (or better than) the results that TOTAL and/or VAXTOTAL provide for these problems.

5.2.2 Program Flow - ICECAP shall be run to determine whether or not the program flows properly. The program must transition to valid known states. The program must not hang in an endless loop. The ability to exit gracefully from the program must be demonstrated.

5.2.3 On-line Assistance - The ability of the intended user to formulate the commands necessary to design and analyze a control system with the on-line assistance that is provided must be demonstrated.

5.2.4 Output - The ability to document a control system design and analysis selectively and conveniently must be demonstrated.

5.3 TESTING PLAN

The following plan shall be used in testing the previously described requirements.

5.3.1 Informal Testing - All program modules shall be thoroughly tested before they are integrated into the main program. As each module is integrated into the main program, the program shall be thoroughly tested to assure that the added function works properly and that no program degradation has taken place. No formal documentation is required for informal testing.

5.3.2 Formal Testing - Formal testing shall be carried out for a representative sample of the entire program. The following paragraphs constitute this formal testing. The samples shall be chosen to demonstrate formally the ability of the program to meet the requirements defined in Chapter 2.

5.3.2.1 Functional Requirements - The following functional requirements shall be tested using relevant test cases from the student handout [30]. Results shall be compared with the results documented in the student handout to see if they are within reasonable tolerances allowed for the differences in the wordlengths of the Cyber (wordlength = 60 bits) and the VAX 11/780 (wordlength = 32 bits). In accordance with Chapter 2, a tolerance of 0.001 shall be considered reasonable. Results that are out of tolerance shall be documented and explained if possible. The functional requirements to be tested are as follows:

- The use of the DEFINE command to input one or more of the following: GTF, HTF, OLTF, CLTF using both the polynomial form and the factored form.

- The use of the FORM command to form one or more of the following:

  - OLTF using GTF and HTF
  - CLTF using OLTF
  - CLTF using GTF and HTF

- The use of the PRINT LOCUS AUTOSCALE command to demonstrate the autoscaling and proper plotting of the Root Locus.

- The use of the PRINT SPECS command to demonstrate the ability to analyze the specifications of the control system properly using a step input.

5.3.2.2 Program Flow - A representative subset of all possible ICECAP commands shall be tested to assure that the program flows properly and does not hang in an endless loop. The following commands shall form the minimum subset:

- DEFINE AMAT
- COPY AMAT BMAT
- PRINT BMAT
- DEFINE GTF FACT
- DEFINE HTF POLY
- FORM OLTF
- PRINT OLTF
- FORM CLTF USING OLTF
- PRINT LOCUS AUTOSCALE
- PRINT LOCUS MAGNIFY
- HELP SYSTEM
- HELP INITIAL
- HELP COPY
- DISPLAY SPECS
- TURN MAINMENU ON
- TURN MAINMENU OFF
- UPDATE
- RECOVER
- STOP

5.3.2.3 On-Line Assistance - This requirement shall be tested in two ways:

5.3.2.3.1 By the use of the HELP command. The information provided shall be reviewed for clarity and usefulness to the user, particularly the novice user.

- o HELP INITIAL
- o HELP SYSTEM
- o HELP COPY

5.3.2.3.2 By typing commands one word at a time followed by a carriage return to determine whether all information is provided to enable the user to formulate an entire ICECAP command. The following words shall be used as starting points for formulating these commands:

- o DEFINE
- o HELP
- o COPY
- o FORM
- o DISPLAY
- o PRINT
- o TURN

5.3.2.4 Output - The ability of the system to provide printouts to document a control system design adequately shall be tested. The results of running the tests on the functional requirements defined above can be used. Printouts shall be produced for the following items:

- o GTF
- o HTF
- o OLTF

- o CLTF
- o a Root Locus
- o a set of specifications

5.4 TESTING RESULTS

The tests of the previous sections were run. Both formal and informal testing were conducted.

5.4.1 Informal Testing - Day to day use of the program as well as thorough testing of each module as it was developed and integrated constituted the informal testing. All of the new and revised code was exercised. Printouts were made and results were compared against hand calculations and against runs of TOTAL and VAXTOTAL. All problem areas uncovered during this testing were either resolved or documented in Appendix C as problem reports. Note that Appendix C also documents problems with TOTAL and VAXTOTAL.

5.4.2 Formal Testing - Appendix J contains the testing documentation generated as a result of the formal testing conducted in accordance with the test plan. This appendix contains copies of screen displays as well as copies of printed output generated when ICECAP was executed. All testing results are satisfactory.

5.5 SUMMARY

The requirements for testing ICECAP have been restated from Chapter 2. The plan for carrying out the various tests has been provided and the testing results and anomolies have been documented. Information on where to find the various testing related documents has been provided.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 INTRODUCTION

This chapter discusses the conclusions reached as a result of having done this thesis effort. The chapter also makes recommendations regarding the continuation of the effort, detailing features that should be implemented.

6.2 CONCLUSIONS

The following conclusions were reached:

6.2.1 It Is Feasible To Interface Pascal With FORTRAN

It is indeed feasible and practical to interface a Pascal Program to FORTRAN modules. The ICECAP main program is written in Pascal and has several modules written in Pascal. FORTRAN subroutines are called by Pascal procedures. There are defined ways on how to call a FORTRAN subroutine from a Pascal program. These ways are described in the VAX/VMS manuals [22, 23, 24, 25].

6.2.2 Modular Structure Is Workable For ICECAP

The use of a highly structured language does allow one to define an overall structure and fill in the detailed functions at a later time. This technique does allow one to have a program that functions properly for those features that one has implemented. Thus several people can work on separate portions of the ICECAP program either separately or together and even at different periods of time during the program development cycle.

6.2.3 CRT Terminals Are Superior To Printing Terminals

ICECAP was tested using several students and instructors as subjects. Without exception, those subjects who had previously used printing terminals for TOTAL felt that the CRT terminal interactive sessions were superior to the printing terminal interactive sessions, especially since printouts of their choosing could be made available at any time during the sessions. The fast response time of the CRT terminal made the session much more pleasant and productive. More iterations of a design problem can be done per unit time using the CRT than can be done using a printing terminal.

6.2.4 ICECAP On-Line Help Is Effective

On-line help, if properly implemented, is more effctive than help that must be obtained from manuals. The fact that there is one fewer book to manage at the terminal is a practical consideration that enhances the productivity of any session, especially where there is very limited terminal space, such as in a school environment.

6.3 RECOMMENDATIONS

The following recommendations are made:

6.3.1 Thesis Effort Be Continued

This thesis effort should be continued. This is a very worthwhile effort. The need for a tool that does complex control system design and analysis has already been established. The need for a tool that is easy and pleasant to use is just now becoming appreciated, since the importance of human engineering is becoming more and more recognized.

6.3.2 More Continuous Time Functions Be Implemented

Although this thesis effort concentrated on implementing the continuous time functions first, not all of these functions were implemented. The functions yet to be implemented include the following:

- o Tabular listing of F(t) (cf. TOTAL Option 31)
- o Plot F(t) at user's terminal (cf. TOTAL Option 32)
- o Printing the time equation (cf. TOTAL Option 35)
- o Partial fraction expansion (cf. TOTAL Option 36)
- o Selection of step, ramp, impulse, pulse or sine input (cf. TOTAL Option 39)

6.3.3 More Root Locus Functions Be Implemented

This thesis effort concentrated on plotting the root locus at the user's terminal and in a print file, adding the automatic scaling feature. Other root locus functions should be added as follows:

- o Zoom feature. With this feature the user would have to enter only a center point of interest (an X coordinate and a Y coordinate) and a positive distance. With that information the program would calculate the borders and display the root locus using those borders.
- o Root locus with a gain of interest (cf. TOTAL Option 42)
- o Root locus with a damping ratio of interest (cf. TOTAL Option 43)
- o List n points on a branch of interest (cf. TOTAL Option 44)

6.3.4 Other Functional Capability Be Implemented

Other capabilities which should be implemented are as follows:

- o Discrete time functions

- Matrix manipulation functions
- Polynomial manipulation functions
- Robustness analyses
  - Return-difference transfer functions
  - Singular value decomposition
  - Plots of max/min singular value vs. frequency

6.3.5 Stochastic Functions Be Defined

Since there is no Stochastic capability either in ICECAP or TOTAL at this time, the desirability for and an approach for adding this capability should be studied. The Stochastic functions recommended for implementation should be very carefully and unambiguously defined. Initially, the study can be limited to the following:

- Time invariant systems
- Stationary noises
- Constant-gain filters
- Constant-gain controllers

These areas of study would lead to an approach for defining the following capabilities:

- Design and analysis of constant-gain LQG controllers
- Design and analysis of constant-gain Kalman filters
- Generation of power spectral density (PSD) plots of any signal of interest

- o Generation of a plot of the autocorrelation kernel as the inverse Fourier transform of that PSD

6.3.6 Teach Modules Be Developed

An approach for developing the teach modules should be carefully studied. The lesson objectives should be outlined and lesson plans should be developed before any attempt is made to implement a teaching module.

6.3.7 Data Dictionary For TOTAL Be Established

Since many of the modules of TOTAL will be the basis for ICECAP, it is important that the baseline of TOTAL be better understood. The variable names in TOTAL are not at all descriptive. It oftentimes takes days, even weeks, to understand some of the modules. Once the effort has been expended in understanding the workings of these modules, that knowledge should be documented so that others do not have to expend the same effort. One of the best ways to document this corporate memory is through a data dictionary for TOTAL, wherein all of the variables and subroutines are explained. The dictionary should be sent to all those who are involved in changing TOTAL for contributions. There should be an OPR (Office of Primary Responsibility) established to collect the contributions and to publish the updated dictionary periodically.

6.3.8 ICECAP Be Rehosted

ICECAP should be rehosted on the AFIT VAX which operates under the UNIX operating system. This system is intended for student class use and can accommodate a larger student load than can the AFIT VAX in the Digital Engineering Laboratory which is intended to be used for research. Once ICECAP is rehosted, the students and faculty can use ICECAP instead of Cyber TOTAL.

6.3.9 Plot Capability Be Enabled

Since it has been determined that the printer in the AFIT Digital Engineering Laboratory now has a working graphics capability, an effort should be made to implement a plotting capability in ICECAP.

6.4 SUMMARY

Several conclusions were reached as a result of this thesis effort. This chapter has detailed these conclusions. This chapter also details several recommendations regarding the continuation of the ICECAP project. Recommendations on what functional capabilities ought to be implemented in ICECAP have been included.

BIBLIOGRAPHY

1. Astrom, K. J. and H. Elmquist. "Perspective on Interactive Software for Computer Aided Modeling and Design of Control Systems," Proceedings of the 20th IEEE Conference on Decision and Control, December 1981.

2. Balchen, Jens G. and Arne Tysso. "Application of CAD in Modeling, Identification and Control of Industrial and Large Scale, Nontechnical Processes," Proceedings of the 20th IEEE Conference on Decision and Control, December 1981.

3. Balfour, A. and D. H. Marwick. Programming in Standard FORTRAN 77. New York, New York: North-Holland, Inc., 1979.

4. Brubaker, Thomas A. Development of Improved Design Methods for Digital Filtering Systems. AFAL-TR-77-207. Fort Collins, Colorado: Colorado State University, 1 November 1977.

5. Colgate, James A. INTERAC - An Interactive Software Package for Direct Digital Control Design. MS Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1977.

6. D'Azzo, John J. and Constantine H. Houpis. Linear Control System Analysis and Design: Conventional and Modern, Second Edition. New York, New York: McGraw-Hill Book Company, 1981.

7. Dakin, Karl J. and David A. Higgins. "Fingerprinting a Program," Datamation, April 1982.

8. Davis, Richard M. Thesis Projects in Science and Engineering. New York, New York: St. Martin's Press, 1980.

BIBLIOGRAPHY

9. Didaleusky, Dennis G. J. Multi-Rate Digital Control Systems with Simulation Applications, Volume III: Source Listings. AFWAL-TR-80-3101. Wright-Patterson Air Force Base, Ohio: Flight Dynamics Laboratory, September 1980.

10. Digital Equipment Corporation. EDT Editor Manual. AA-J726A-TC. Maynard, Massachussetts: Digital Equipment Corporation, October 1980.

11. Digital Equipment Corporation. Engineering Systems Software Referral Catalog, Sixth Edition. Maynard, Massachusetts: Digital Equipment Corporation, 1980.

12. Digital Equipment Corporation. VAX Systems and Options Summary. Maynard, Massachussetts: Digital Equipment Corporation, April 1981.

13. Digital Equipment Corporation. VAX/VMS Command Language User's Guide. AA-D023C-TE. Maynard, Massachusetts: Digital Equipment Corporation, May 1982.

14. Digital Equipment Corporation. VAX/VMS Guide to Using Command Procedures. AA-H782B-TE. Maynard, Massachusetts: Digital Equipment Corporation, May 1982.

15. Digital Equipment Corporation. VAX/VMS Primer. Maynard, Massachusetts: Digital Equipment Corporation, 1978.

16. Digital Equipment Corporation. VAX/VMS Software Handbook. Maynard, Massachusetts: Digital Equipment Corporation, 1978.

17. Digital Equipment Corporation. VAX/VMS Summary Description. Maynard, Massachusetts: Digital Equipment Corporation, 1978.

18. Digital Equipment Corporation. VAX-11 DIGITAL Standard Runoff (DSR) User's Guide. AA-J268B-TK. Maynard, Massachusetts: Digital Equipment Corporation, May 1982.

19. Digital Equipment Corporation. VAX-11 FORTRAN Language Reference Manual. AA-D034C-TE. Maynard, Massachusetts: Digital Equipment Corporation, April 1982.

20. Digital Equipment Corporation. VAX-11 FORTRAN Language Reference Manual. AA-D034B-TE. Maynard, Massachusetts: Digital Equipment Corporation, April 1980.

BIBLIOGRAPHY

21. Digital Equipment Corporation. VAX-11 FORTRAN User's Guide. AA-D035C-TE. Maynard, Massachusetts: Digital Equipment Corporation, April 1982.

22. Digital Equipment Corporation. VAX-11 Pascal Language Reference Manual. AA-H484B-TE. Maynard, Massachusetts: Digital Equipment Corporation, March 1981.

23. Digital Equipment Corporation. VAX-11 Pascal Language Reference Manual. Maynard, Massachusetts: Digital Equipment Corporation, November 1979.

24. Digital Equipment Corporation. VAX-11 Pascal User's Guide. AA-H485B-TE. Maynard, Massachusetts: Digital Equipment Corporation, March 1981.

25. Digital Equipment Corporation. VAX-11 Pascal User's Guide. Maynard, Massachusetts: Digital Equipment Corporation, November 1979.

26. Digital Equipment Corporation. VAX-11 Run-Time Library User's Guide. AA-L824A-TE. Maynard, Massachusetts: Digital Equipment Corporation, April 1982.

27. Digital Equipment Corporation. VAX-11 Run-Time Library Reference Manual. AA-D036C-TE. Maynard, Massachusetts: Digital Equipment Corporation, April 1982.

28. Digital Equipment Corporation. VAX-11 Symbolic Debugger Reference Manual. AA-D026C-TE. Maynard, Massachusetts: Digital Equipment Corporation, January 1979.

29. Digital Equipment Corporation. VT-100 User Guide. Second Edition. Maynard, Massachusetts: Digital Equipment Corporation, January 1979.

30. Fontana, Robert E. Sample TOTAL Printouts. Unpublished Student Handouts for Course EE 562. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology.

31. Harvey, C. A. and J. E. Wall. "Phases in the Development of Control System Design Software," Proceedings of the 20th IEEE Conference on Decision and Control, December 1981.

32. Herget, C. J. and Thomas P. Weis. Linear Systems Analysis Program User's Manual. UCID-30184. Livermore, California: Lawrence Livermore Laboratory, October 1980.

BIBLIOGRAPHY

33. Hernandez, Enrique G. An Interactive Computational Aerodynamics Analysis Program. MS Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1980.

34. Holt, R. C. and J. N. P. Hume. Programming Standard Pascal. Reston, Virginia: Reston Publishing Company, Inc., 1980.

35. Houpis, C. H. and Gary B. Lamont. Lecture Notes on Digital Control Systems/Information Processing. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, August 1980.

36. Kennedy, Thomas A. The Design of Digital Controllers for the C-141 Aircraft Using Entire Eigenstructure Assignment and the Development of an Inter-Active Computer Design Program. MS Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, March 1979.

37. Koffman, Elliot B. Problem Solving and Structured Programming in Pascal. Reading, Massachusetts: Addison-Wesley Publishing Company, 1981.

38. Larimer, Stanley J. An Interactive Computer-Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, March 1978.

39. Larimer, Stanley J. TOTAL User's Manual (CAD). Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, June 1981.

40. Larimer, Stanley J. and Holly L. Emrick. "On the Possibility of Common Control System Design Software for Government, Education, and Industry," Unpublished.

41. Logan, Glen T. Development of an Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, March 1982.

42. Mancini, Anthony J. Computer Aided Control System Design Using Frequency Domain Specifications. Monterey, California: Naval Postgraduate School, June 1976.

43. Maybeck, Peter S. Stochastic Models, Estimation and Control. Volume 1. New York, New York: Academic Press, 1979.

BIBLIOGRAPHY

44. Maybeck, Peter S. Stochastic Models, Estimation and Control. Volume 2. New York, New York: Academic Press, 1982.

45. Maybeck, Peter S. Stochastic Models, Estimation and Control. Volume 3. New York, New York: Academic Press, 1982.

46. McQuay, William K. A Comparative Guide to NOS/BE and VAX-11/780 Command Languages. Wright-Patterson Air Force Base, Ohio: Avionics Laboratory, Air Force Wright Aeronautical Laboratories, October 1980.

47. Moynihan, John A. "What Users Want," Datamation, April 1982.

48. Munro, Neil. "Illustration of the Applicability of Computer Aided Design Packages," Proceedings of the 20th IEEE Conference on Decision and Control, December 1981.

49. Musick, Stanton H. SOFE: A Generalized Digital Simulation for Optimal Filter Evaluation User's Manual, AFWAL-TR-80-1108. Wright-Patterson Air Force Base, Ohio: Avionics Laboratory, October 1980.

50. Musick, Stanton H. SOFEPL: A Plotting Postprocessor for 'SOFE' User's Manual, AFWAL-TR-80-1109. Wright-Patterson Air Force Base, Ohio: Avionics Laboratory, November 1981.

51. Nadler, Gerald and Ali Seireg. "Professional Engineering Education in the Classroom," Engineering Education, May 1982.

52. O'Brien, Frederick L. A Consolidated Computer Program for Control System Design. MS Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1976.

53. Polak, E. "Interactive Software for Computer-Aided-Design of Control Systems via Optimization," Proceedings of the 20th IEEE Conference on Decision and Control, December 1981.

54. Prather, Ronald E. Problem Solving Principles: Programming with Pascal. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1982.

55. Shneiderman, Ben. "How to Design with the User in Mind," Datamation, April 1982.

BIBLIOGRAPHY

56. Simpson, Henry. "A Human-Factors Style Guide for Program Design," BYTE, April 1982.

57. Smith, H. T. and T. R. G. Green. Human Interaction with Computers. New York, New York: Academic Press, 1980.

58. Spielman, Stephen Christie. Frequency Methods in Computer Aided Design of Control Systems. MS Thesis. Urbana, Illinois: University of Illinois, Urbana, December 1976.

59. Strang, Gilbert. Linear Analysis and Its Applictions, (Second Edition). New York, New York: Academic Press, 1980.

60. Tiberghien, Jacques. The Pascal Handbook. Berkeley, California: Sybex, 1981.

61. Tausworthe, Robert C. Standardized Development of Computer Software. Englewood Cliffs, New Jersey: Prentice-Hall, 1979.

62. Vines, Larry Paul. Computer Aided Design of Systems. Monterey, California: Naval Postgraduate School, June 1975.

63. Walker, Robert, Charles Gregory, Jr., and Sunil Shah. "MATRIXx: A Data Analysis, System Identification Control Design and Simulation Program," Abstract of paper awaiting publication.

64. Weinberg, Victor. Structured Analysis. New York, New York: Yourdon Press, 1979.

65. Whitbeck, Richard F. and Dennis G. J. Didaleusky. Multi-Rate Digital Control Systems with Simulation Applications, Volume I: Technical Report AFWAL-TR-80-3101. Wright-Patterson Air Force Base, Ohio: Flight Dynamics Laboratory, September 1980.

66. Yourdon, Edward and Larry L. Constantine. Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design. New York, New York: Yourdon Press, 1978.

APPENDIX A

STRUCTURE CHARTS

A.1 INTRODUCTION

This appendix contains the Structure Charts that show all of the calls of the ICECAP main program through and including the subprogram (TOTICE) that interfaces with the VAXTOTAL modules. The VAXTOTAL modules have already been documented in previous works [38, 41] and therefore are not repeated here. This appendix details the standards which were used in developing the structure charts. Appendix B explains the purpose of each module shown in the structure charts of Appendix A.

A.2 STRUCTURE CHART STANDARDS

For the sake of clarity and consistency the following standards were used in developing the structure charts contained in this appendix:

A.2.1 Only one level of depth is portrayed on a given chart.

A.2.2 Open arrows show data flow.

A.2.3 Closed arrows show control that affects program flow.

A.2.4 Diagrams are in alphabetical order.

A.2.5 The extra horizontal line on certain boxes indicates that there are lower level diagrams for those boxes.

A.2.6 The subordinate boxes in each diagram are in the order in which the subprograms are first called. Each subprogram is shown only once even though it may be called more than once.

A.3 ICECAP MODULE HIERARCHY

The following outline shows the hierarchy of the ICECAP modules. A change in a level of indentation indicates that the more deeply indentured module is declared in the less deeply indentured module. For example, Procedure TOTINI is declared in Progam ICER.

STRUCTURE CHARTS

```
Program ICER
   Procedure TOTINI
   Procedure TOTICE
   Procedure FIND_BORDERS
   Procedure MAGNIFY
   Procedure SHRINK
   Function STR$UPCASE
   Procedure CLEAR
   Procedure GRAPHICS
   Procedure NOGRAPHICS
   Procedure HIGHLIGHT
   Procedure NOHIGHLIGHT
   Procedure PAUSE
   Procedure CURSORRC
      Procedure LIB$SET_CURSOR
   Procedure BOXIT
   Procedure TITLE_SLIDE
   Procedure TRIM
   Procedure HELP_PROMPT
   Procedure DICTIONARY
   Procedure READCOM
   Procedure PRINT_BUFFER
   Procedure PACK_BUFFER
   Procedure INTERPRET
      Procedure COPY
      Procedure DEFINE
         Procedure DEFINE_TF
            Procedure DEF_TF_PLANE
         Procedure DEFINE_PROMPT
      Procedure DISPLAY_OR_PRIN
         Procedure LOCUS
            Procedure LOCUS_AUTOSCALE
            Procedure LOCUS_MAGNIFY
            Procedure LOCUS_SHRINK
            Procedure LOCUS_ZOOM
         Procedure FORM
         Procedure HELP
            Procedure HELP_COPY
            Procedure HELP_INITIAL
            Procedure HELP_SYSTEM
      Procedure TURN
         Procedure TURN_X
         Procedure TURN_PROMPT
```

A.4 LIST OF STRUCTURE CHARTS

BOXIT
COPY
DEFINE
DEFINE_PROMPT
DEFINE_TF
DEF_TF_PLANE
DICTIONARY
DISPLAY_OR_PRINT
FORM
HELP
HELP_COPY
HELP_INITIAL
HELP_PROMPT
HELP_SYSTEM
ICECAP
INTERPRET
LOCUS
LOCUS_AUTOSCALE
LOCUS_MAGNIFY
LOCUS_SHRINK
PAUSE
READCOM
TITLE_SLIDE
TURN
TURN_PROMPT
TURN_X

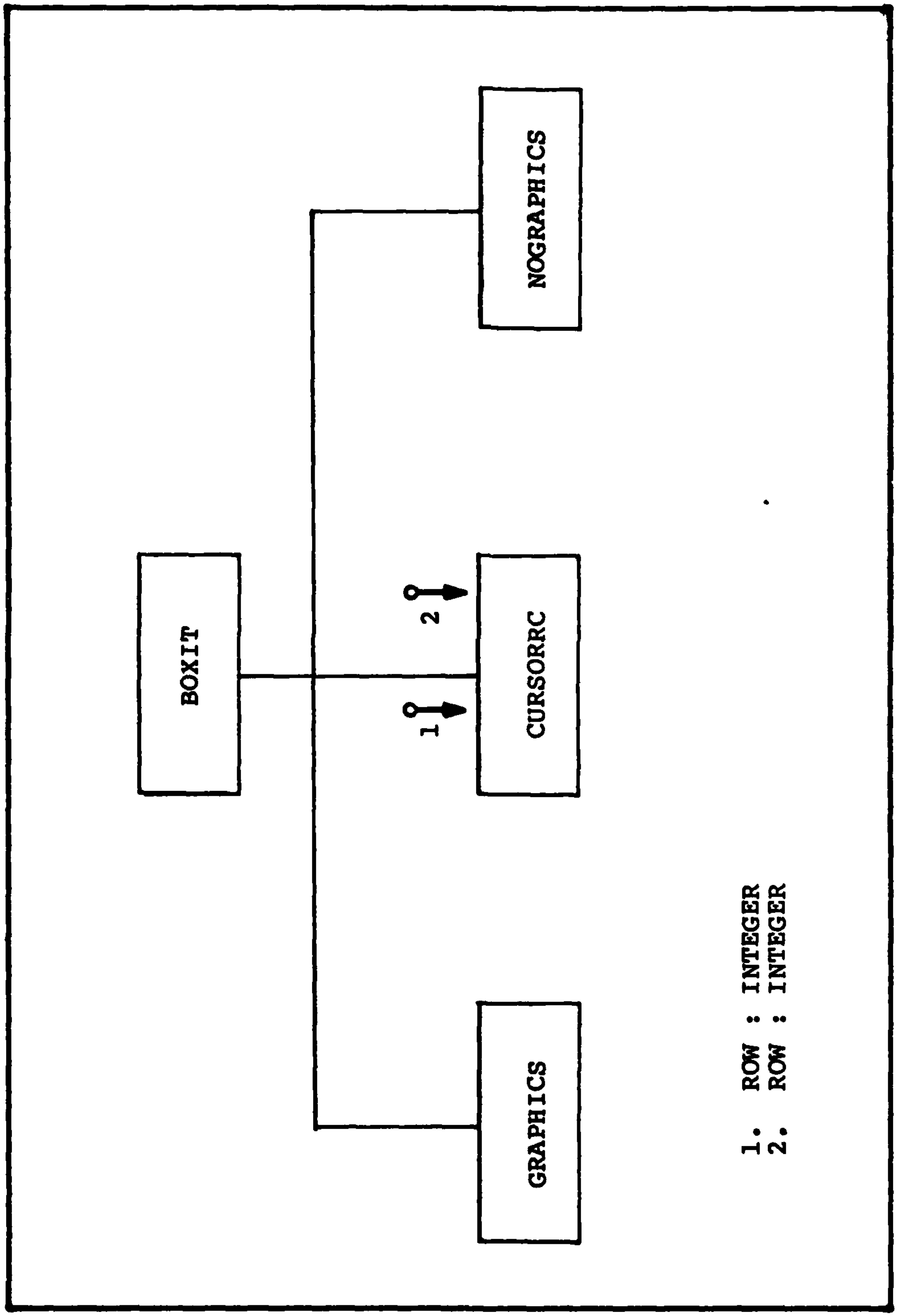


Figure A-1. Structure Chart for Procedure BOXIT

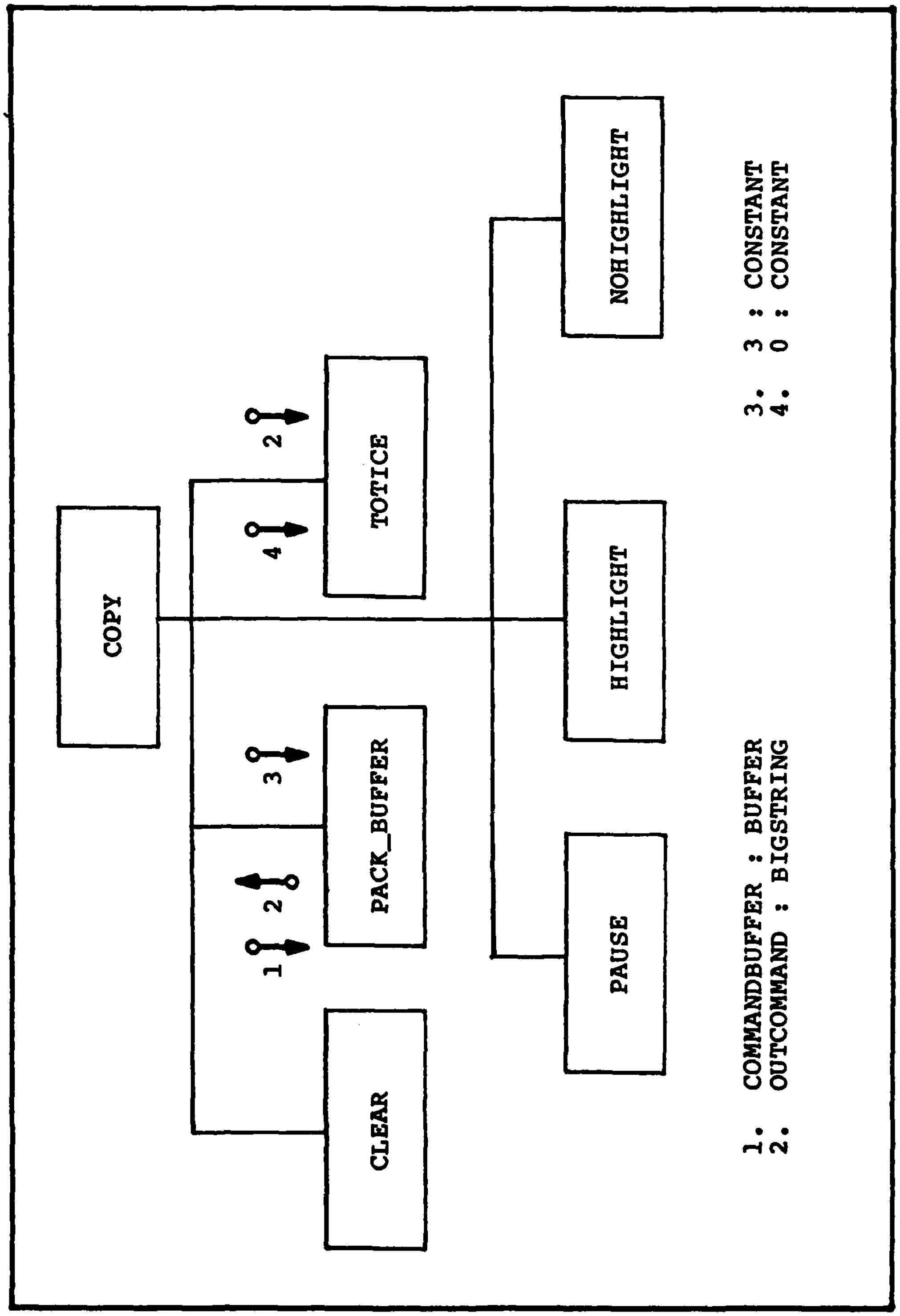


Figure A-2. Structure Chart for Procedure COPY

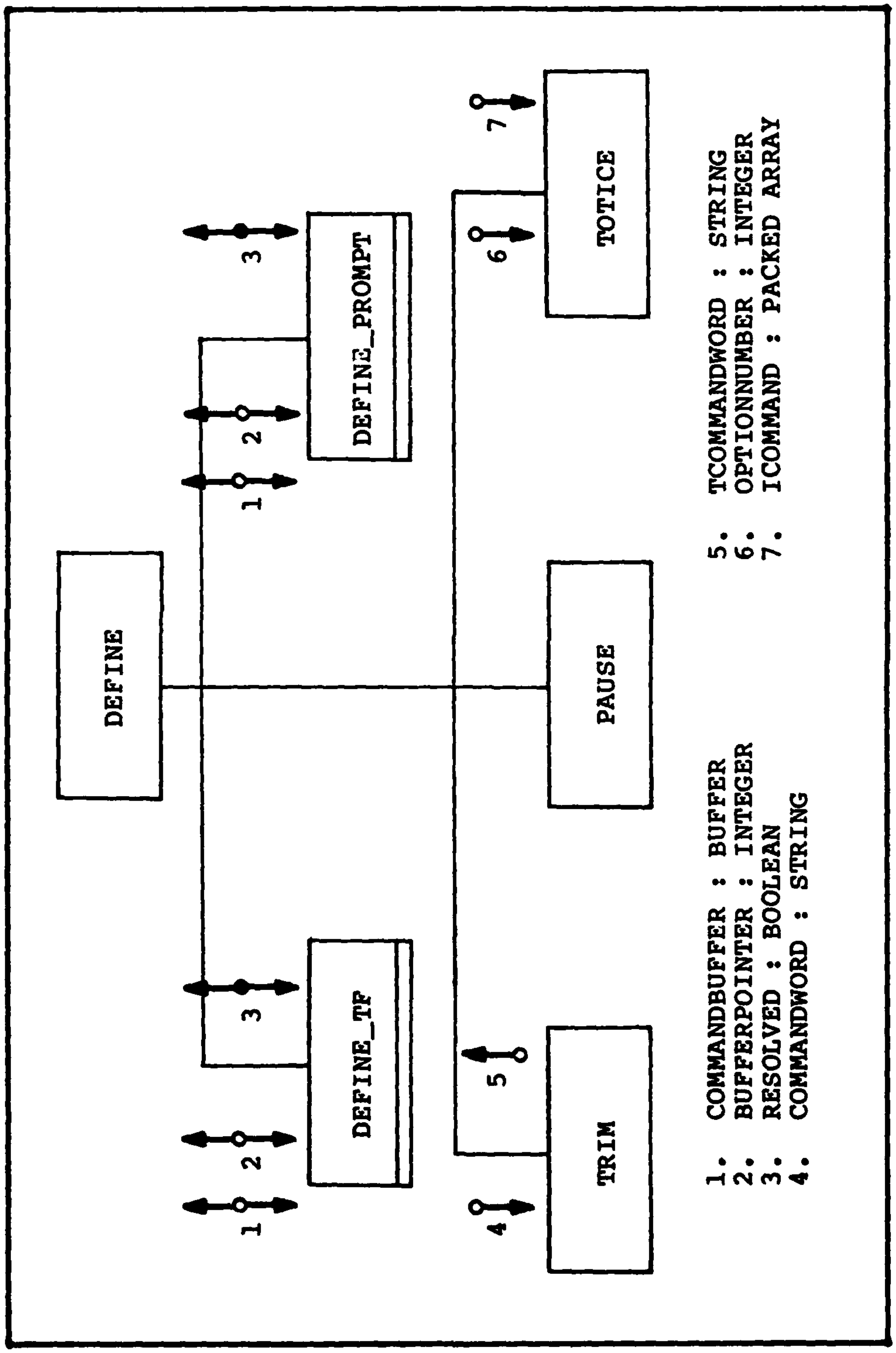


Figure A-3. Structure Chart for Procedure DEFINE

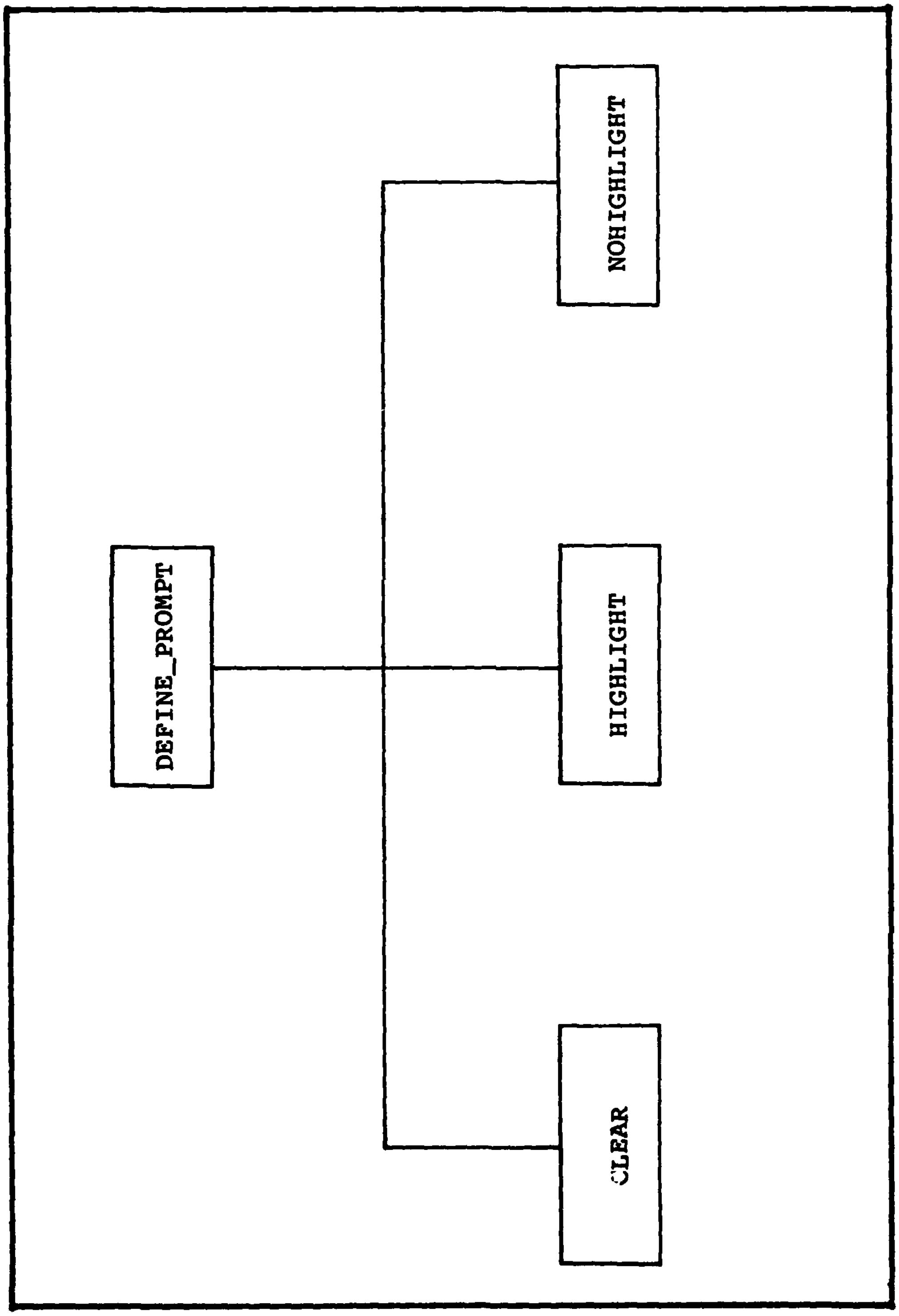


Figure A-4. Structure Chart for Procedure DEFINE_PROMPT

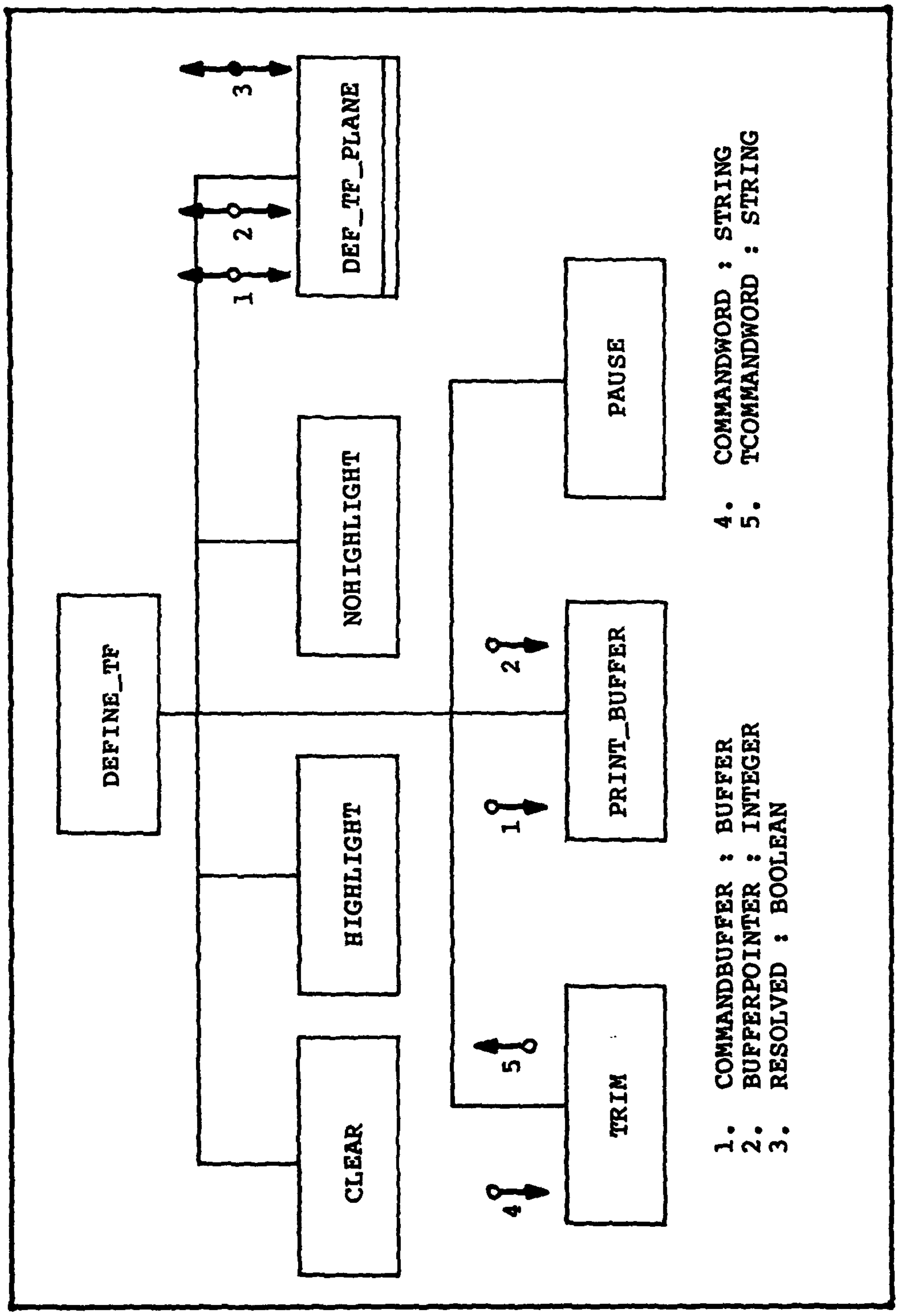


Figure A-5. Structure Chart for Procedure DEFINE_TF

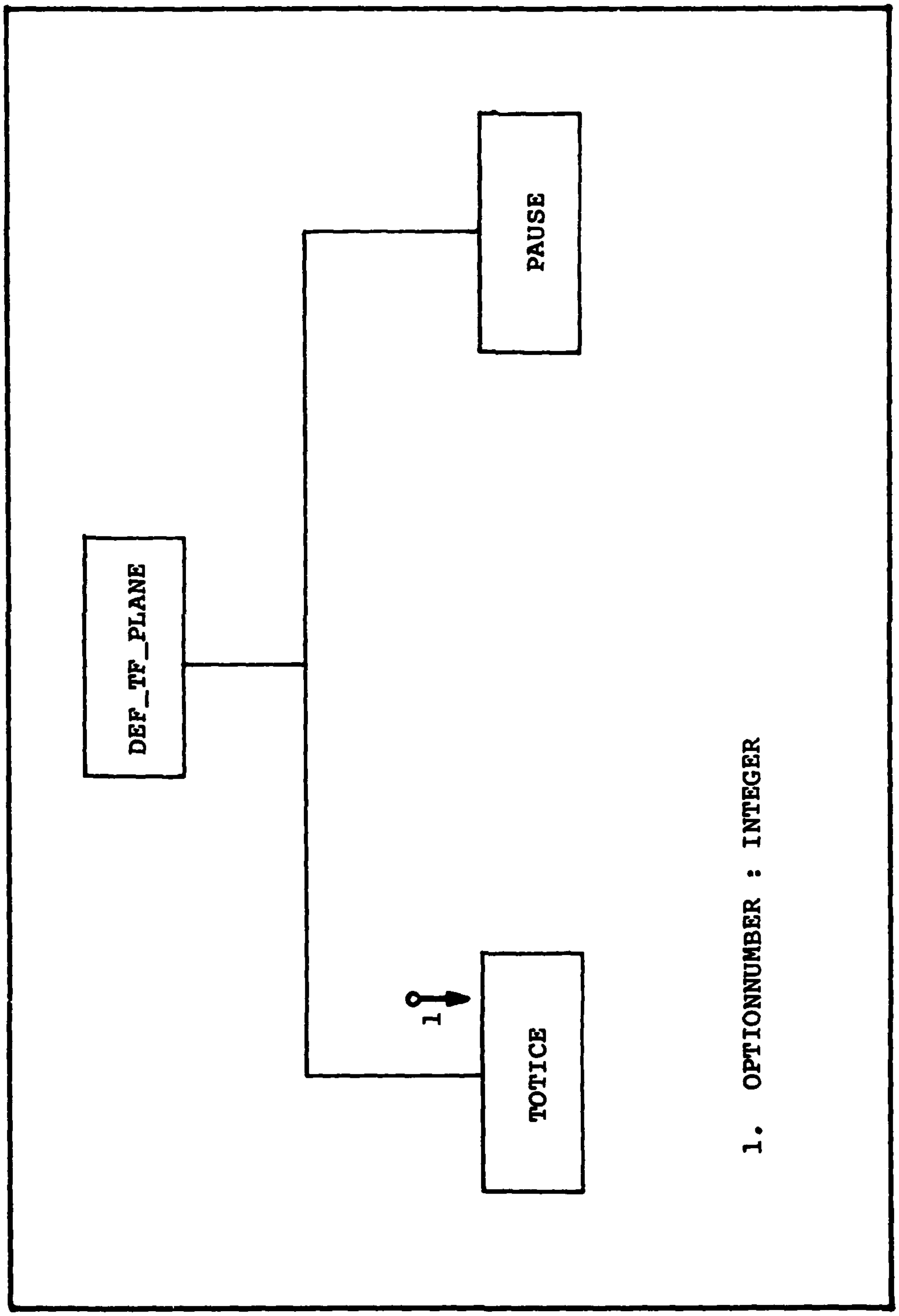


Figure A-6. Structure Chart for Procedure DEF_TF_PLANE

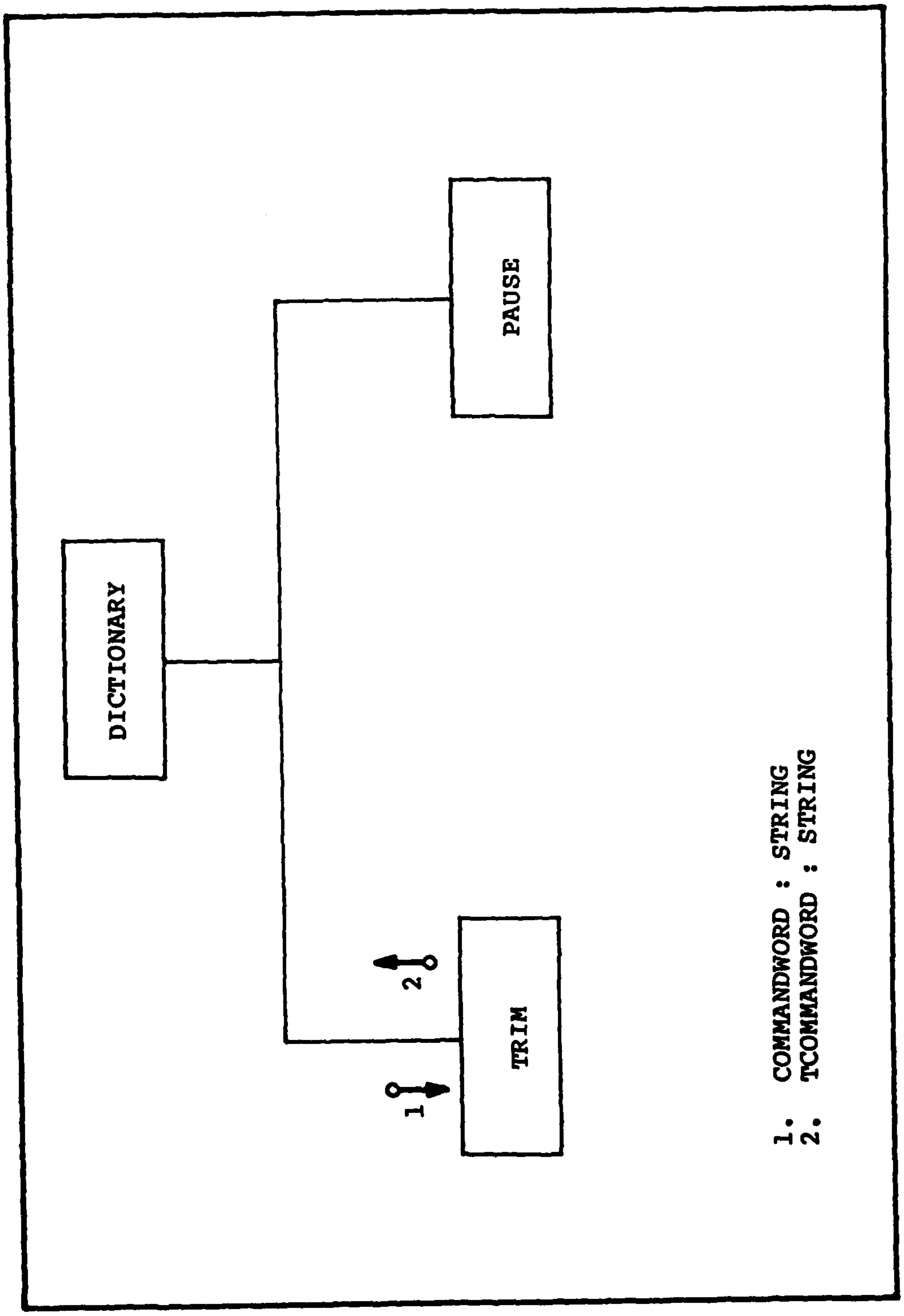
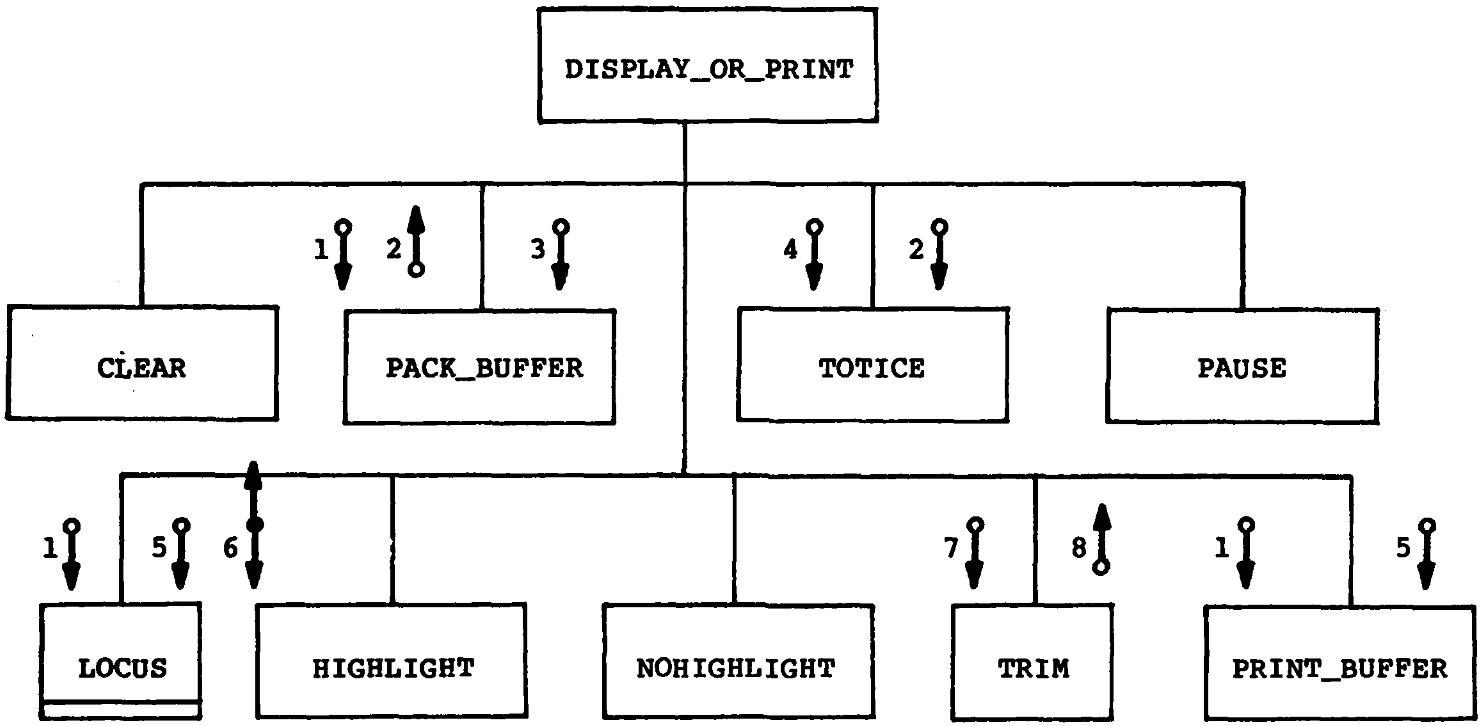


Figure A-7. Structure Chart for Procedure DICTIONARY



1. COMMANDBUFFER : BUFFER
2. OUTCOMMAND : BIGSTRING
3. 2 : CONSTANT
4. OPTIONNUMBER :INTEGER
5. BUFFERPOINTER : INTEGER
6. RESOLVED : BOOLEAN
7. SOURCE : STRING
8. DESTINATION : STRING

Figure A-8. Structure Chart for Procedure DISPLAY_OR_PRINT

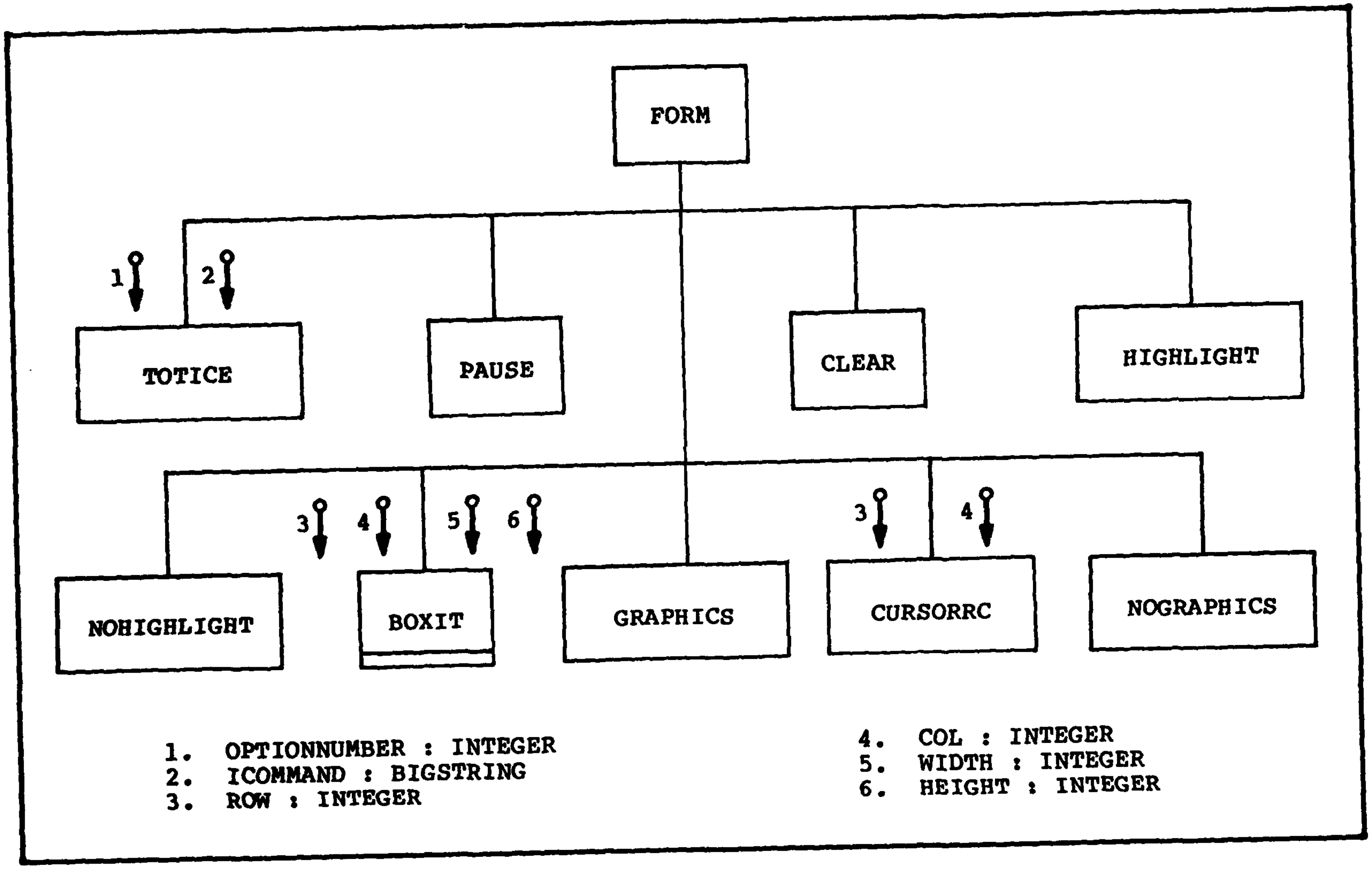


Figure A-9. Structure Chart for Procedure FORM

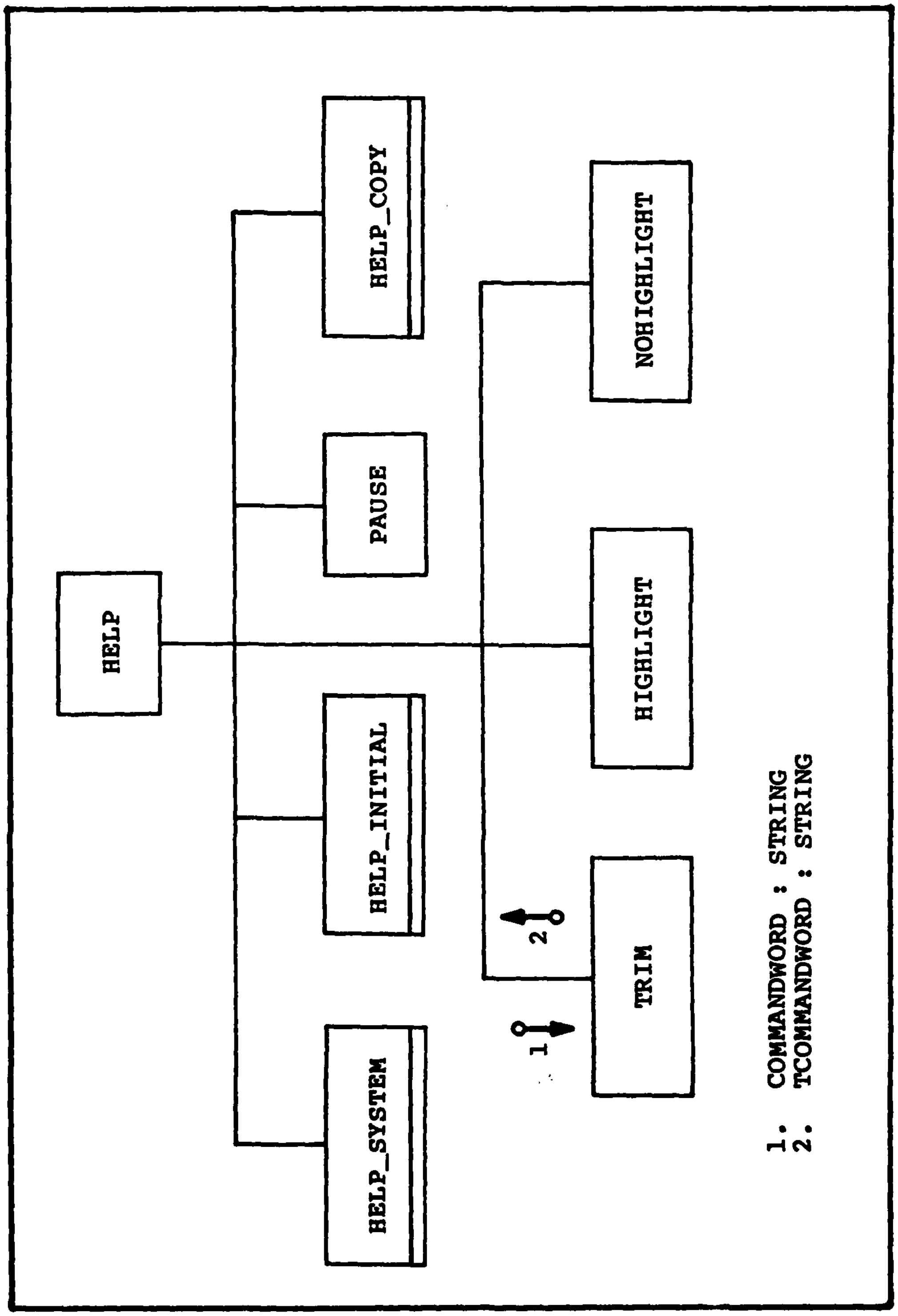


Figure A-10. Structure Chart for Procedure HELP

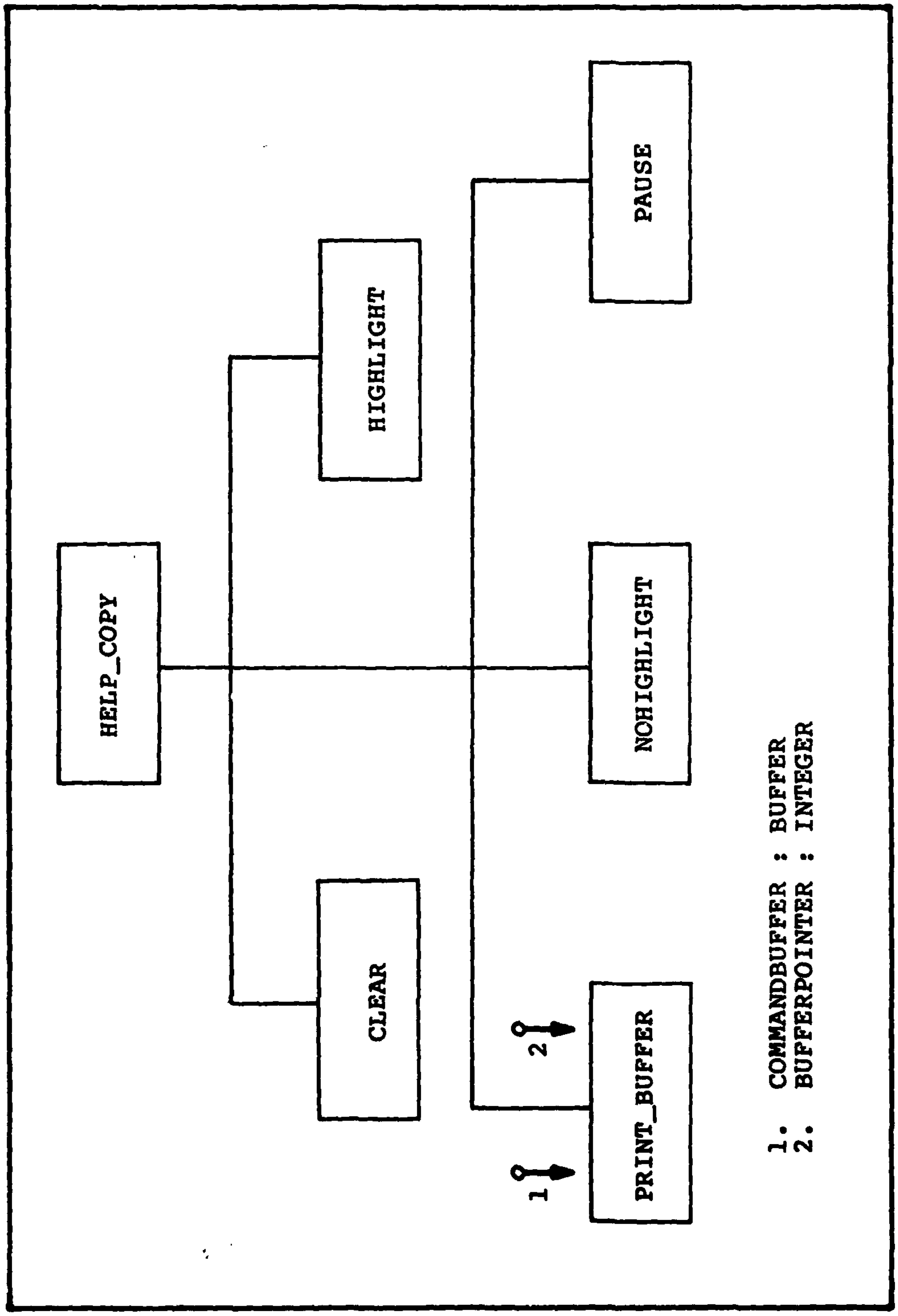


Figure A-11. Structure Chart for Procedure HELP_COPY

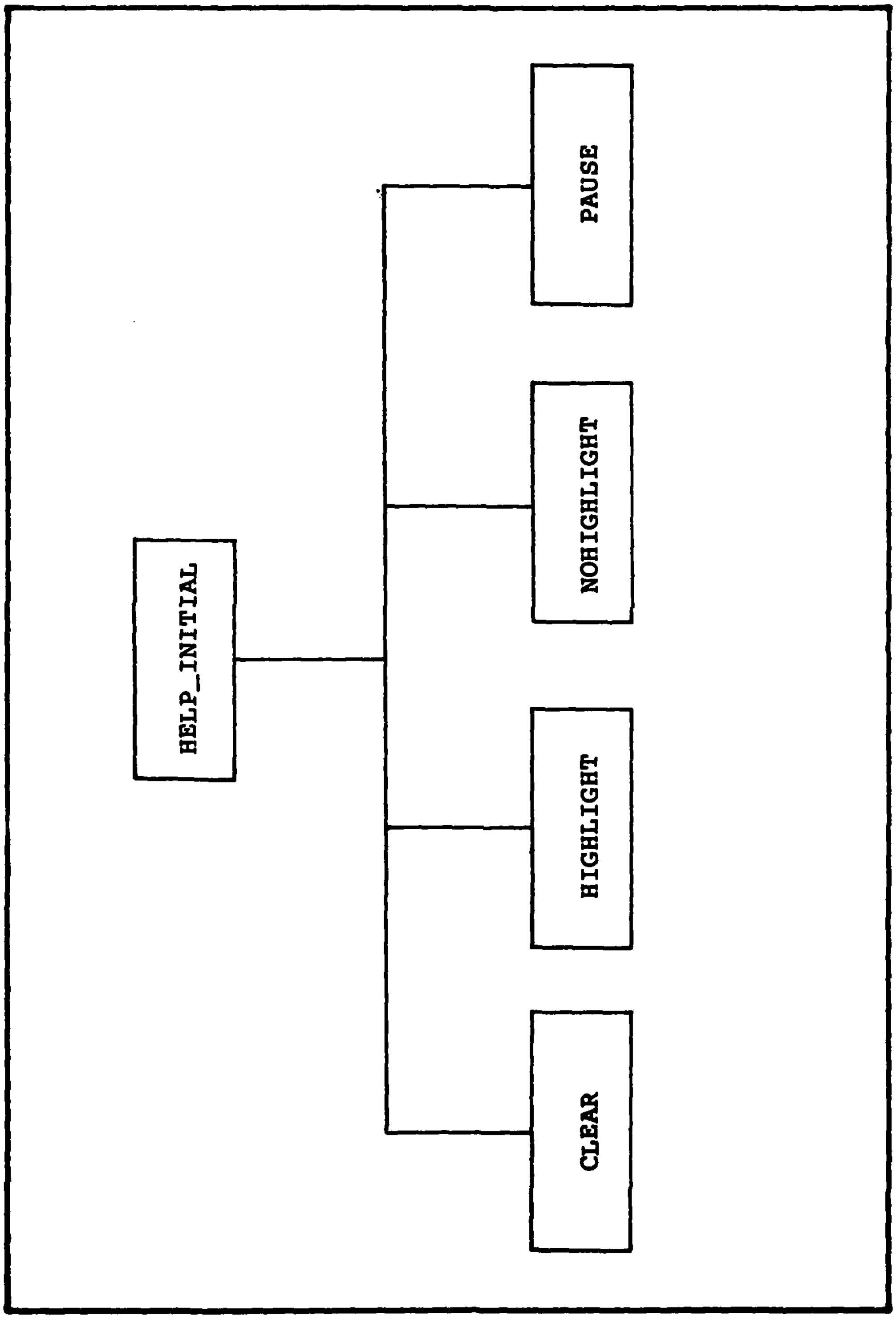


Figure A-12. Structure Chart for Procedure HELP_INITIAL

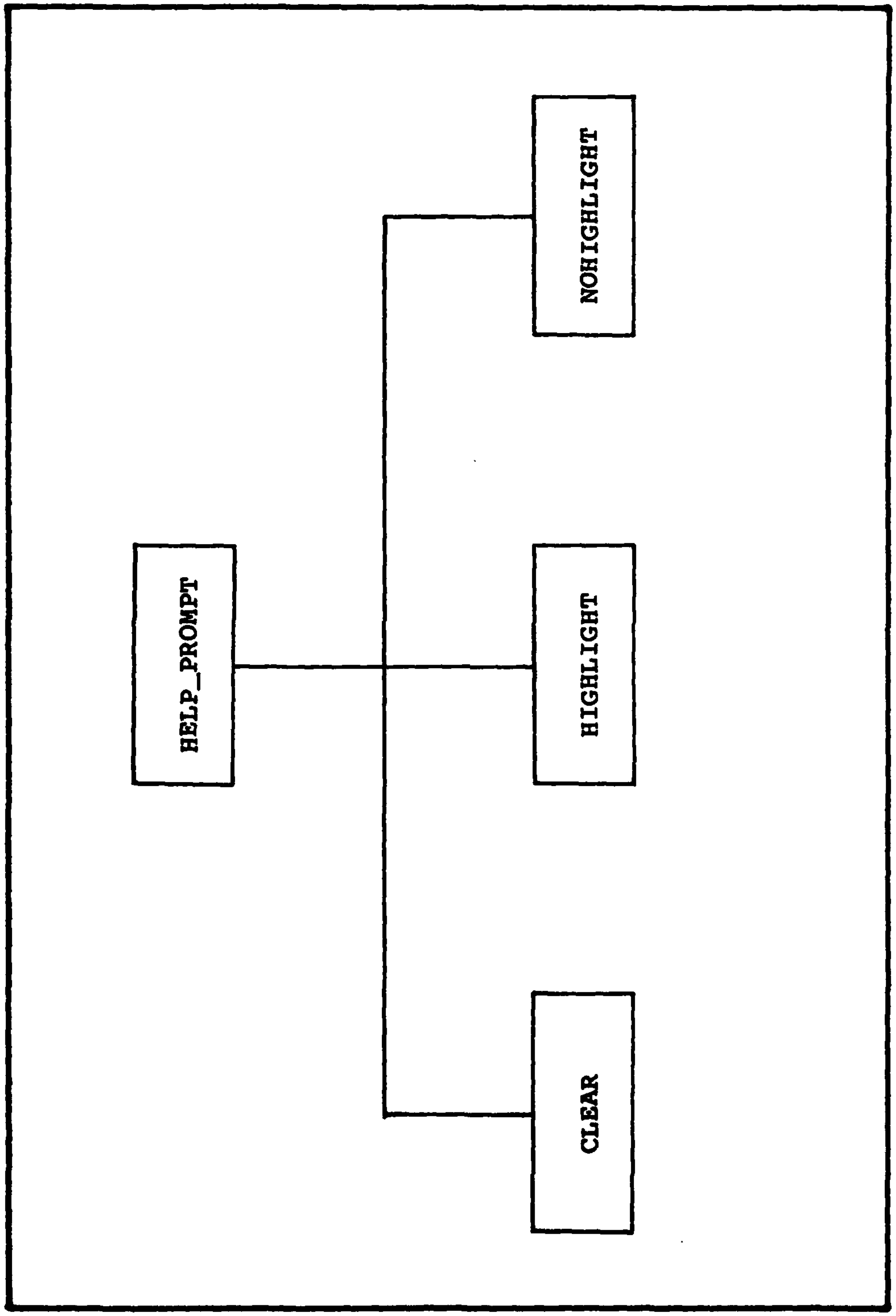


Figure A-13. Structure Chart for Procedure HELP_PROMPT

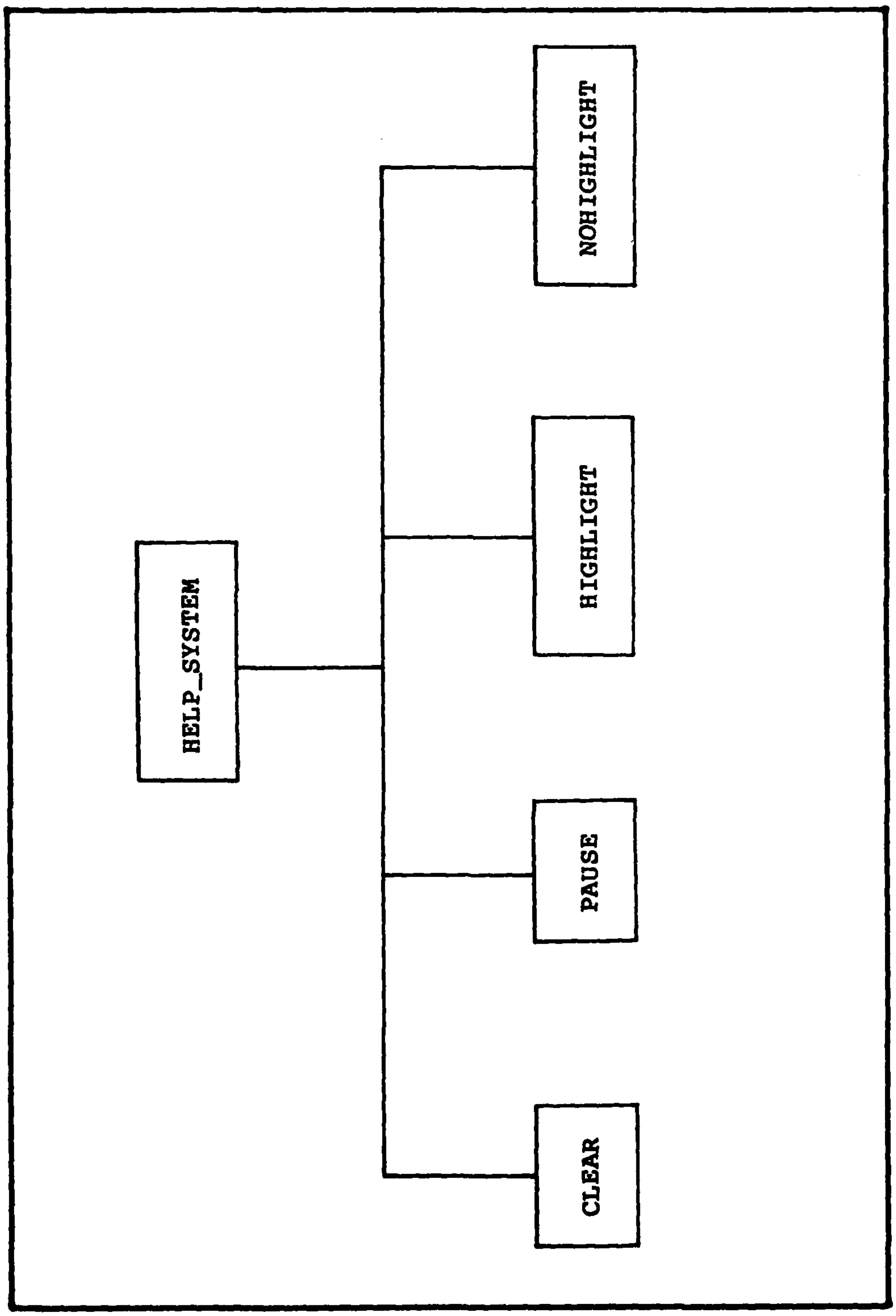


Figure A-14. Structure Chart for Procedure HELP_SYSTEM

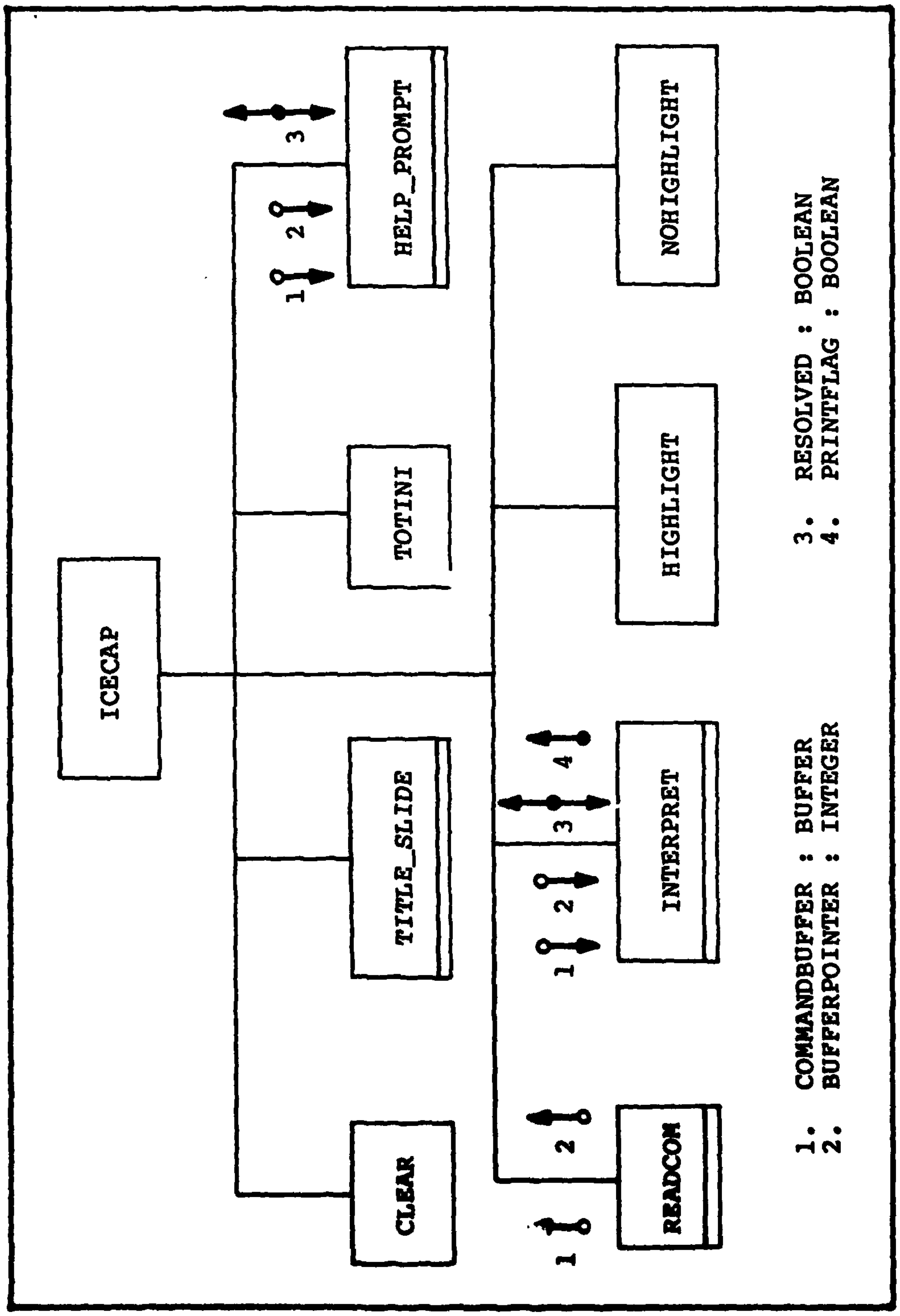


Figure A-15. Structure Chart for Program ICECAP

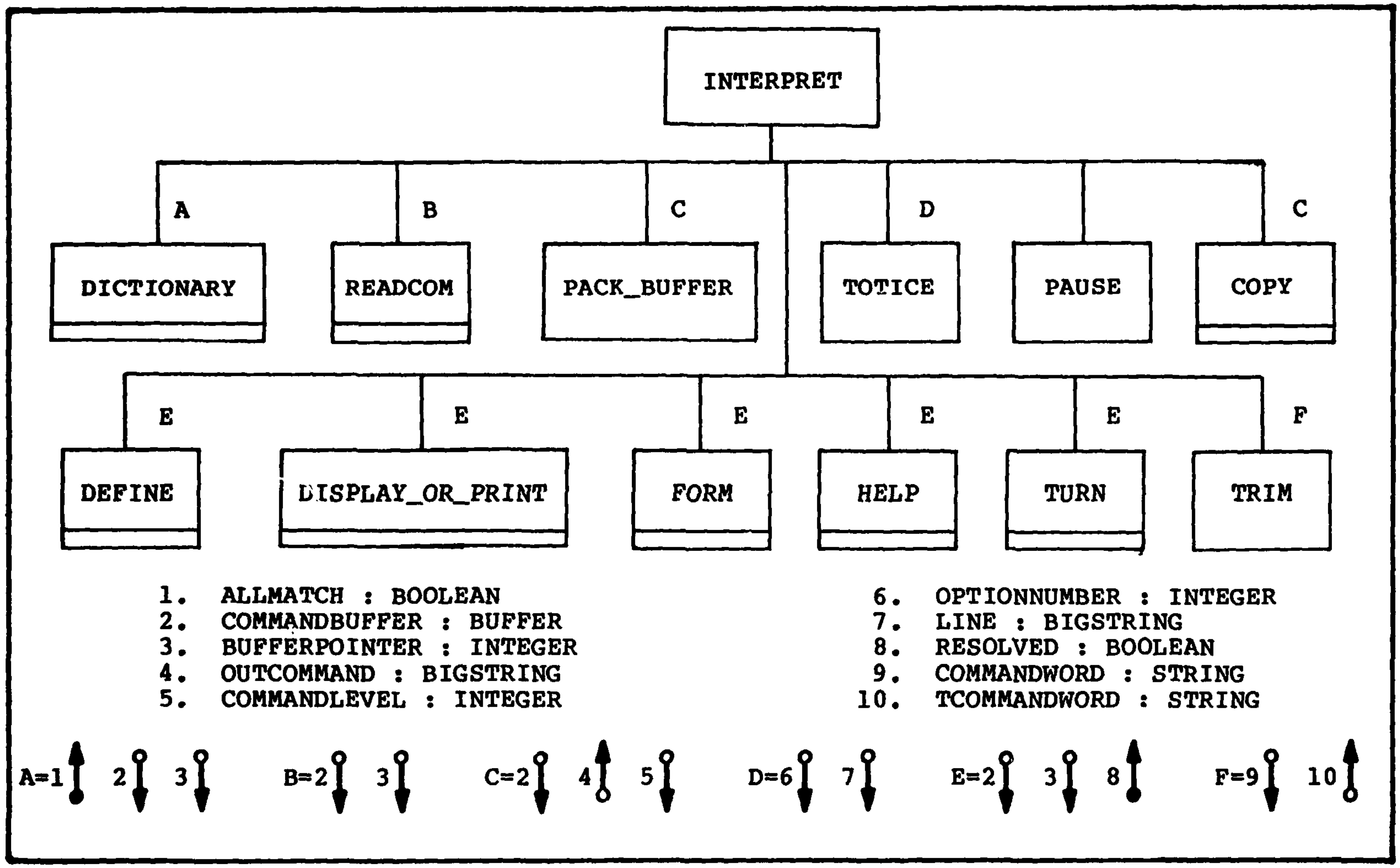


Figure A-16. Structure Chart for Procedure INTERPRET

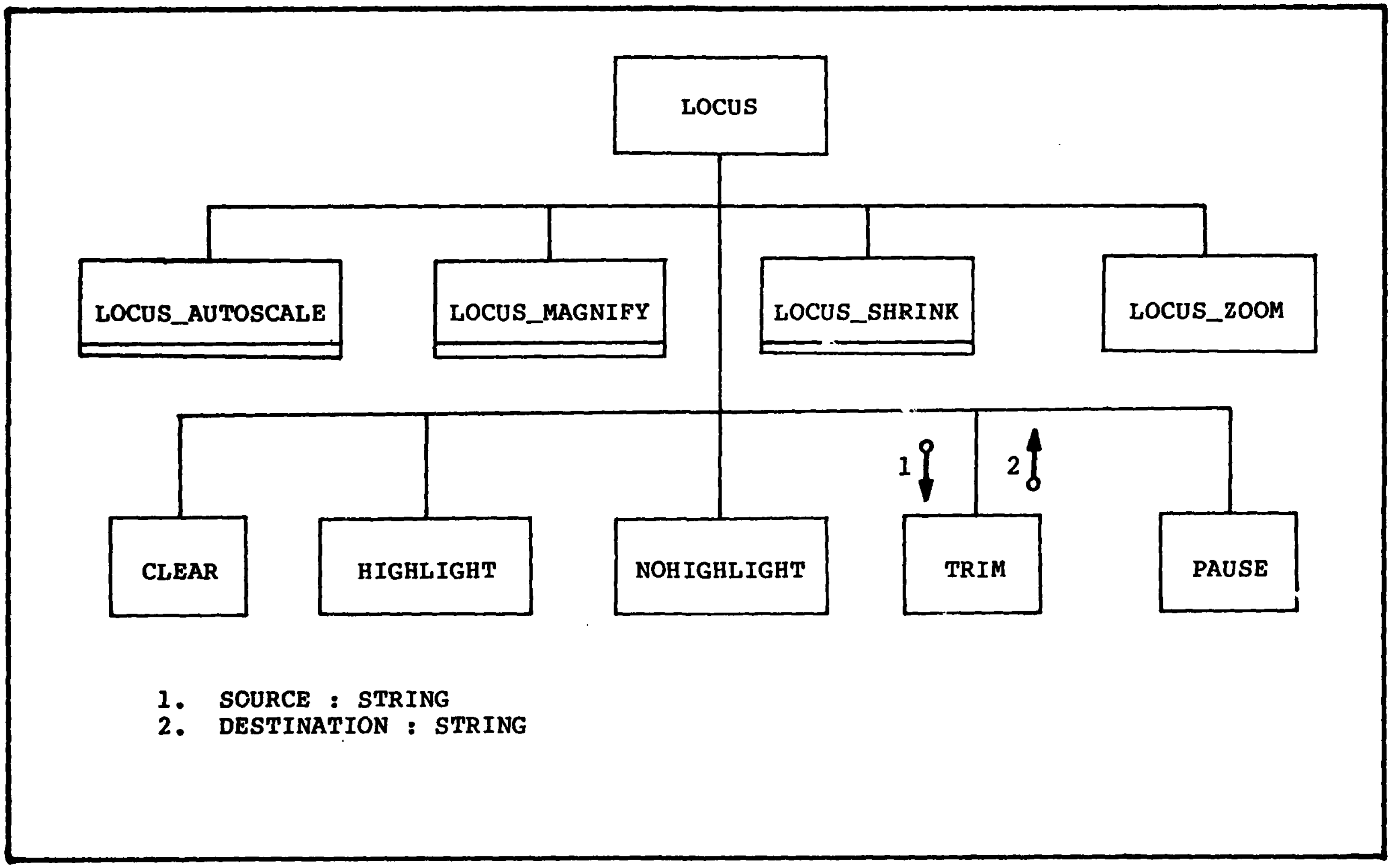
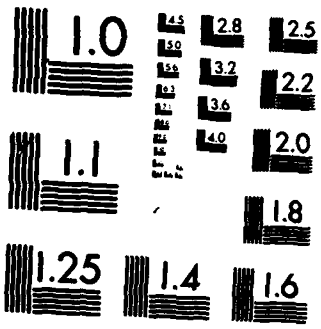


Figure A-17. Structure Chart for Procedure LOCUS

AD-A124 707 DEVELOPMENT OF AN INTERACTIVE CONTROL ENGINEERING COMPUTER ANALYSIS PACKA..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 2/2

UNCLASSIFIED C J GEMBAROWSKI DEC 82 F/G 9/2 NL

END
DATE
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

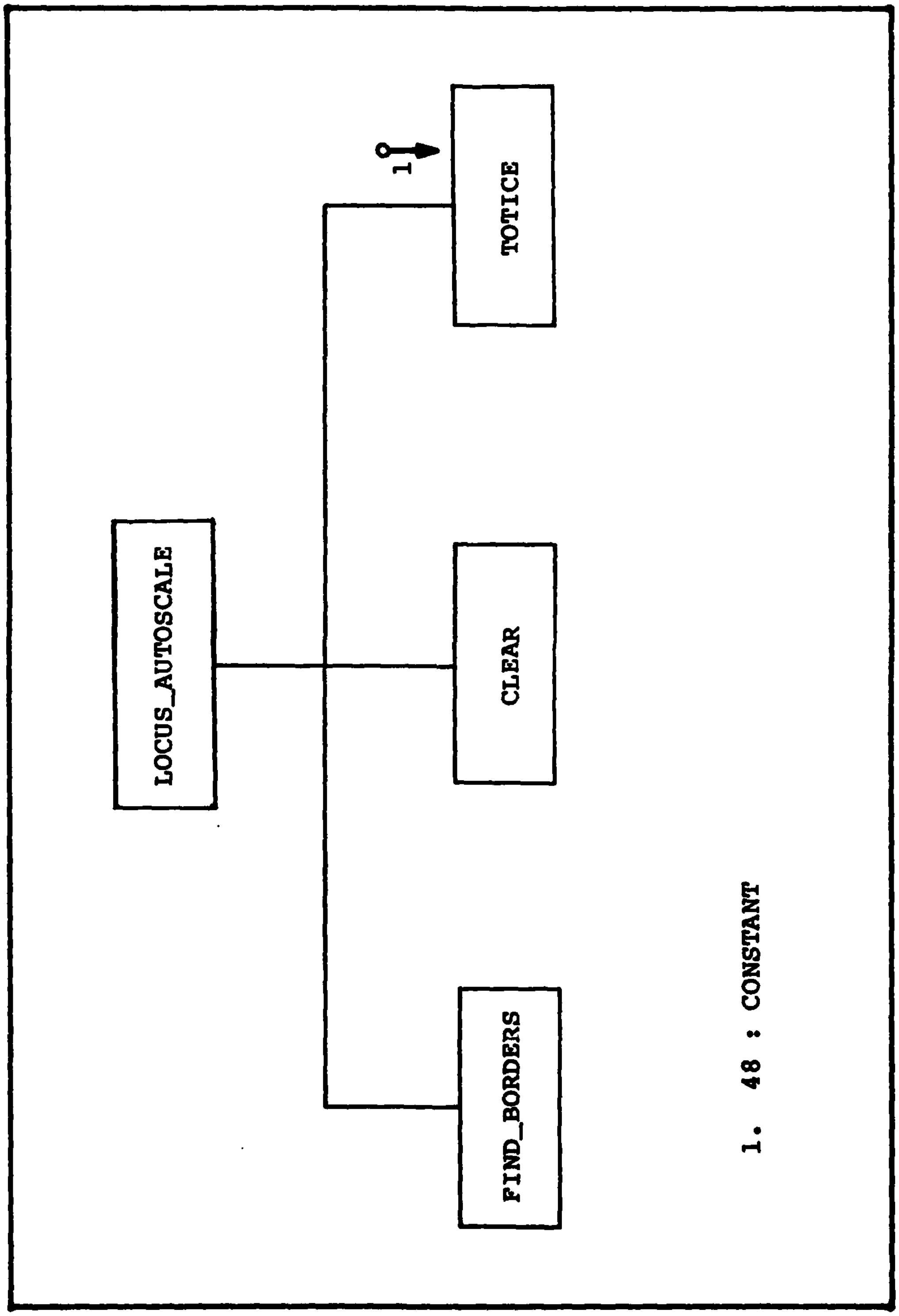


Figure A-18. Structure Chart for Procedure LOCUS_AUTOSCALE

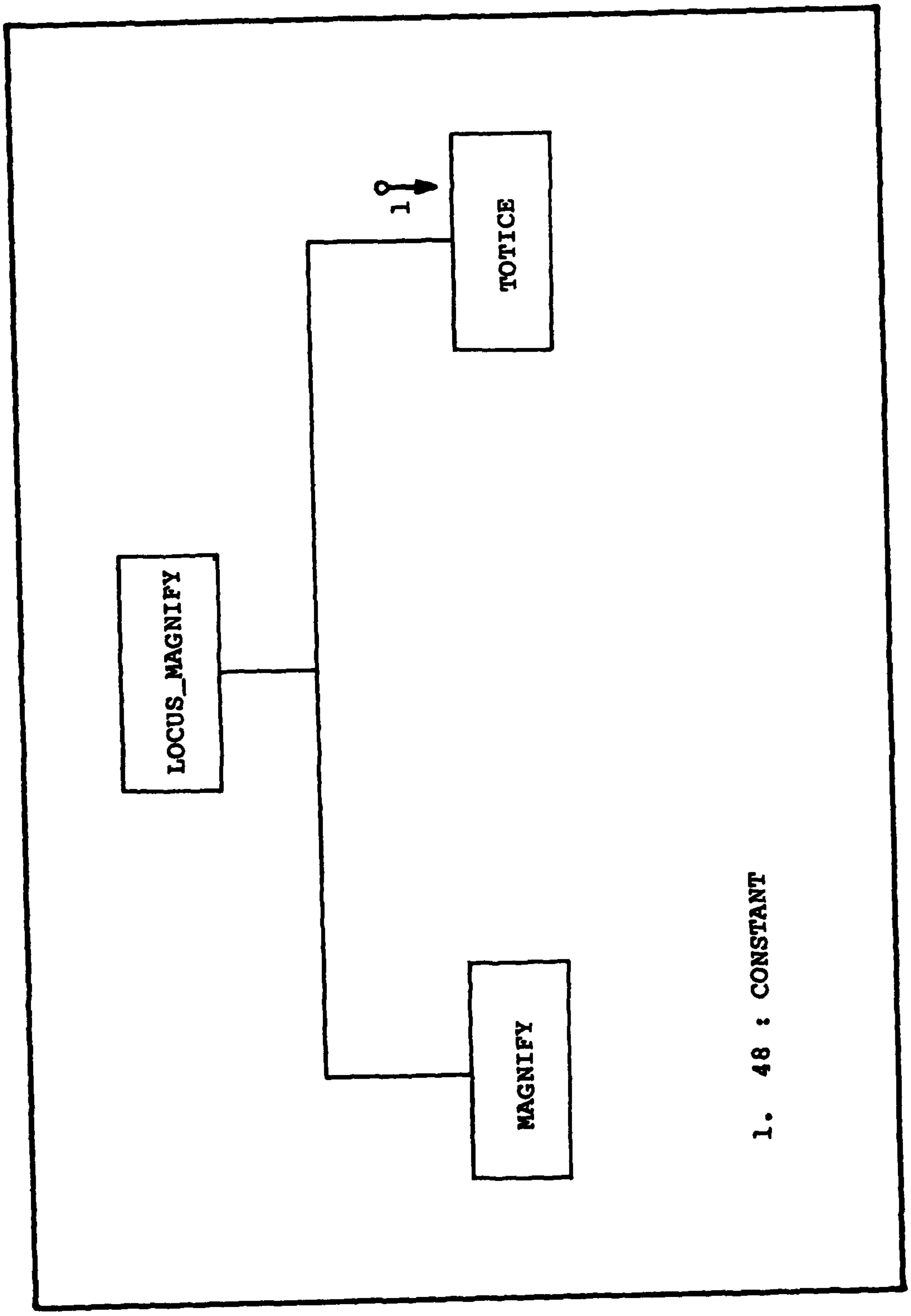


Figure A-19. Structure Chart for Procedure LOCUS_MAGNIFY

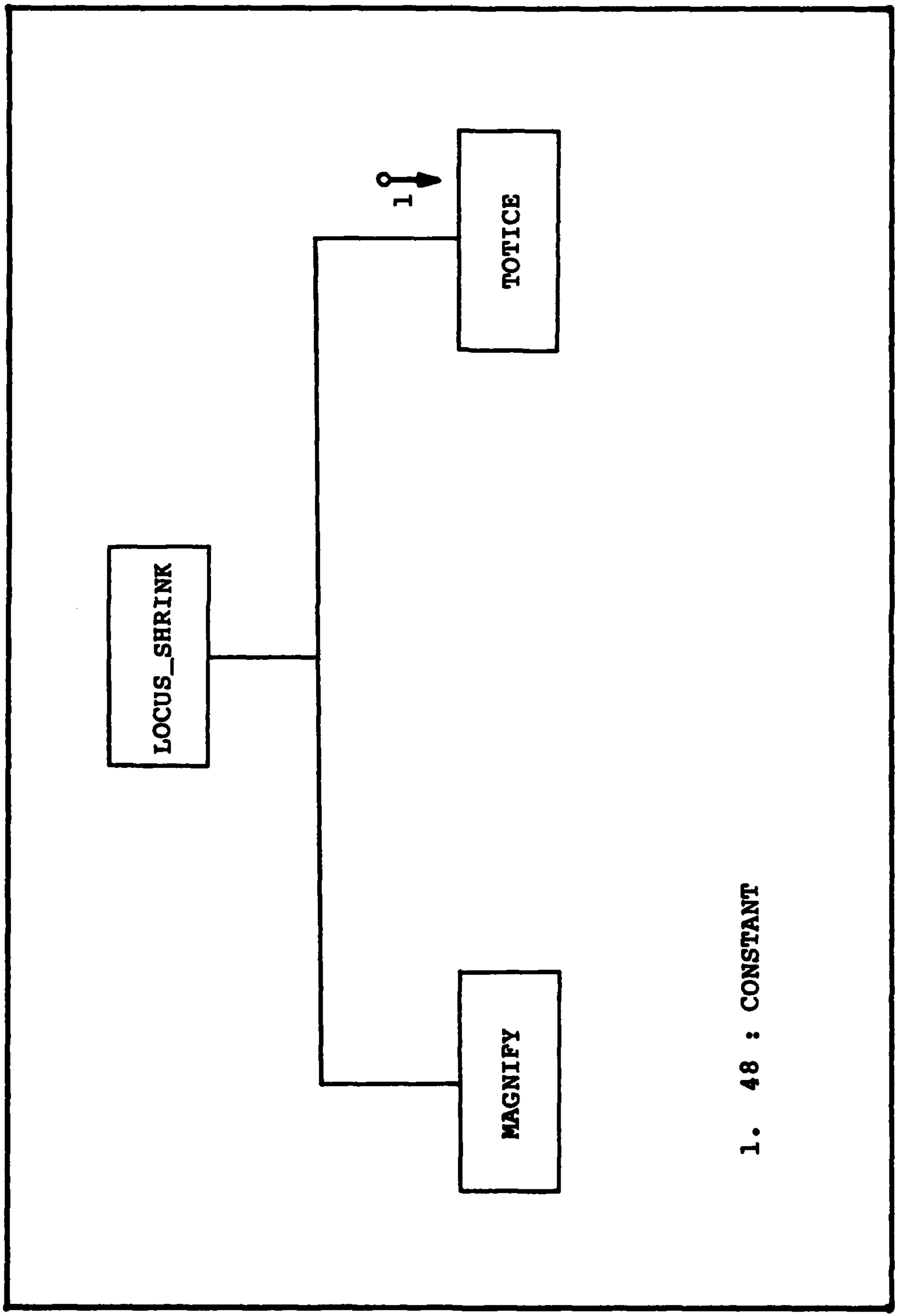


Figure A-20. Structure Chart for Procedure LOCUS_SHRINK

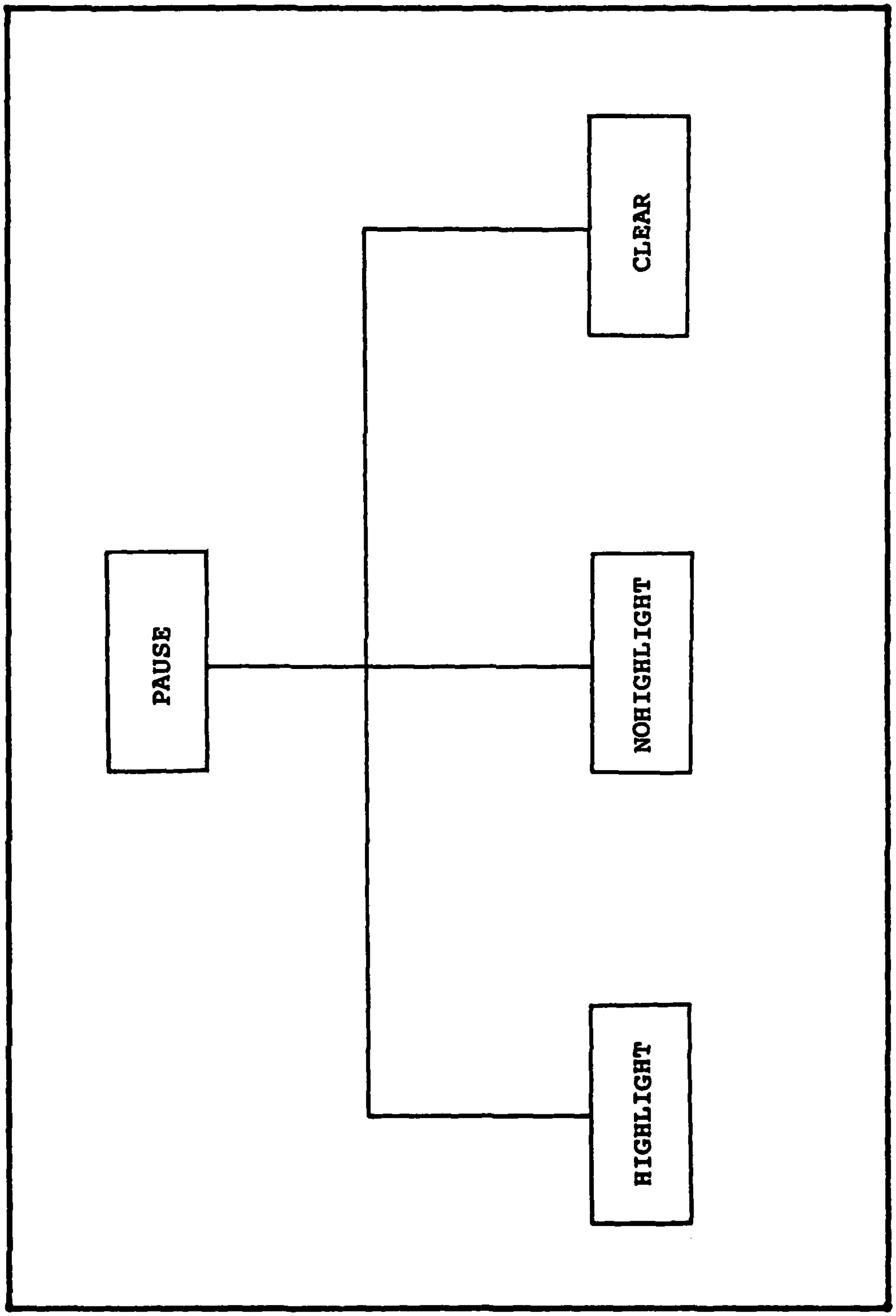


Figure A-21. Structure Chart for Procedure PAUSE

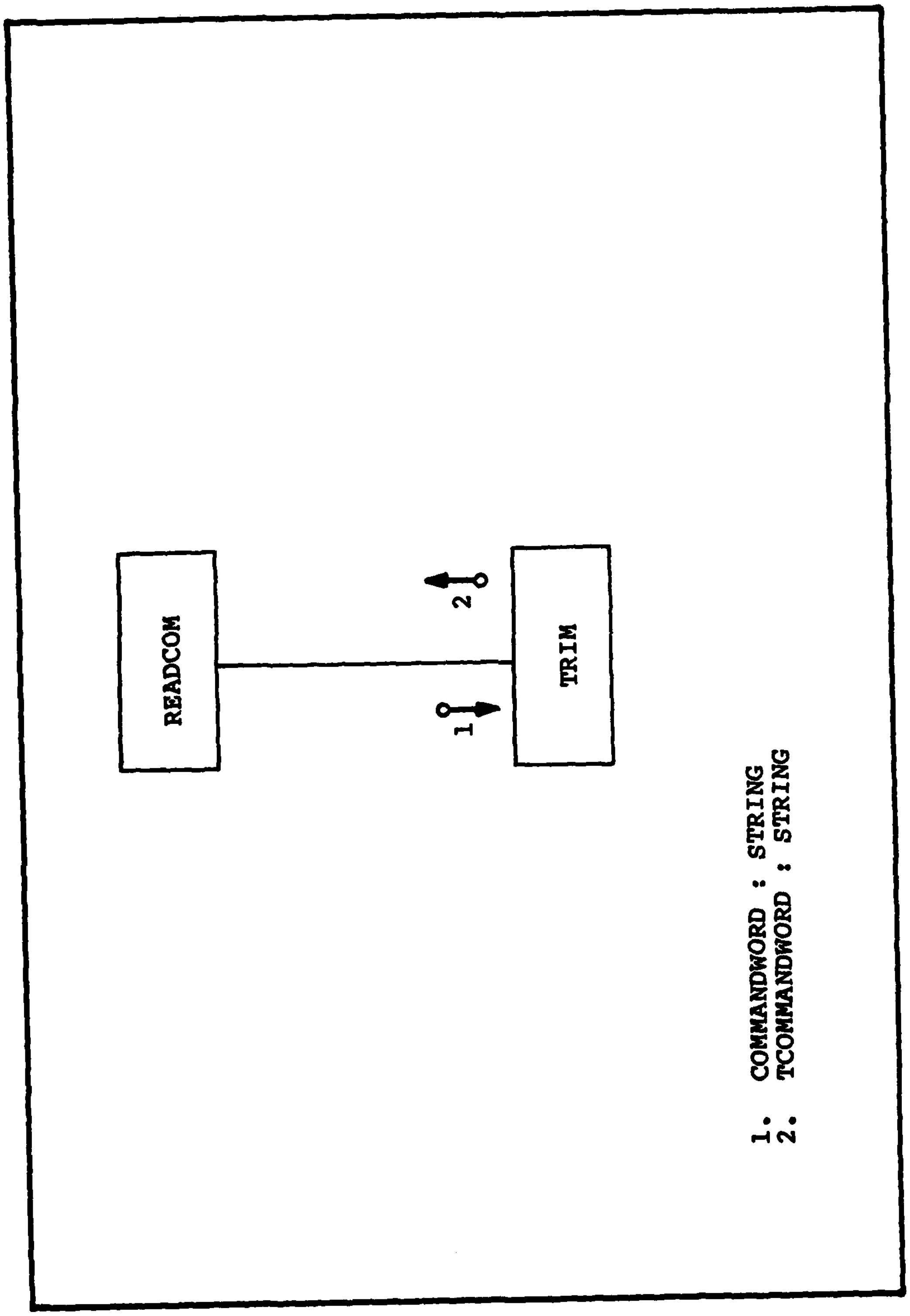


Figure A-22. Structure Chart for Procedure READCOM

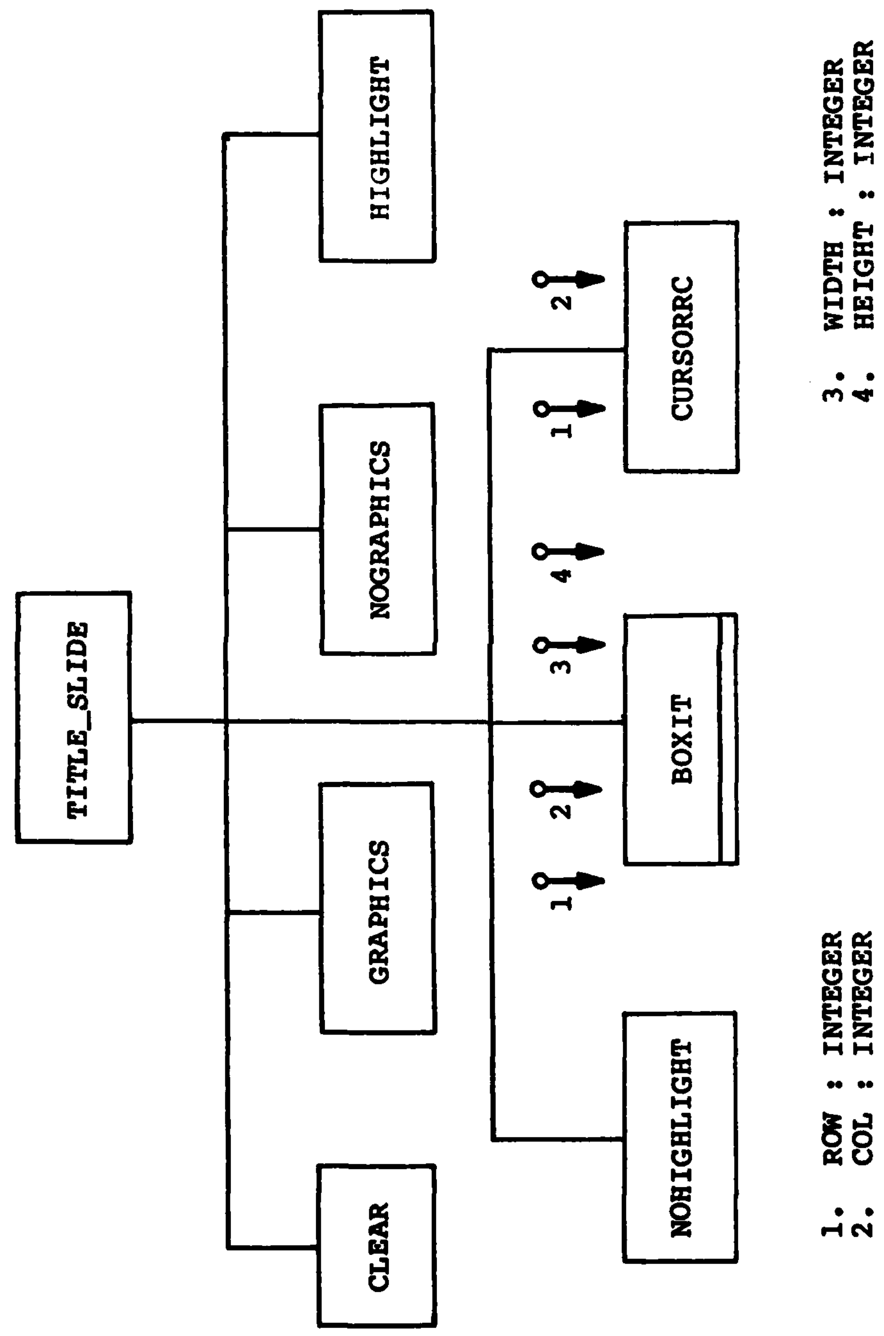


Figure A-23. Structure Chart for Procedure TITLE_SLIDE

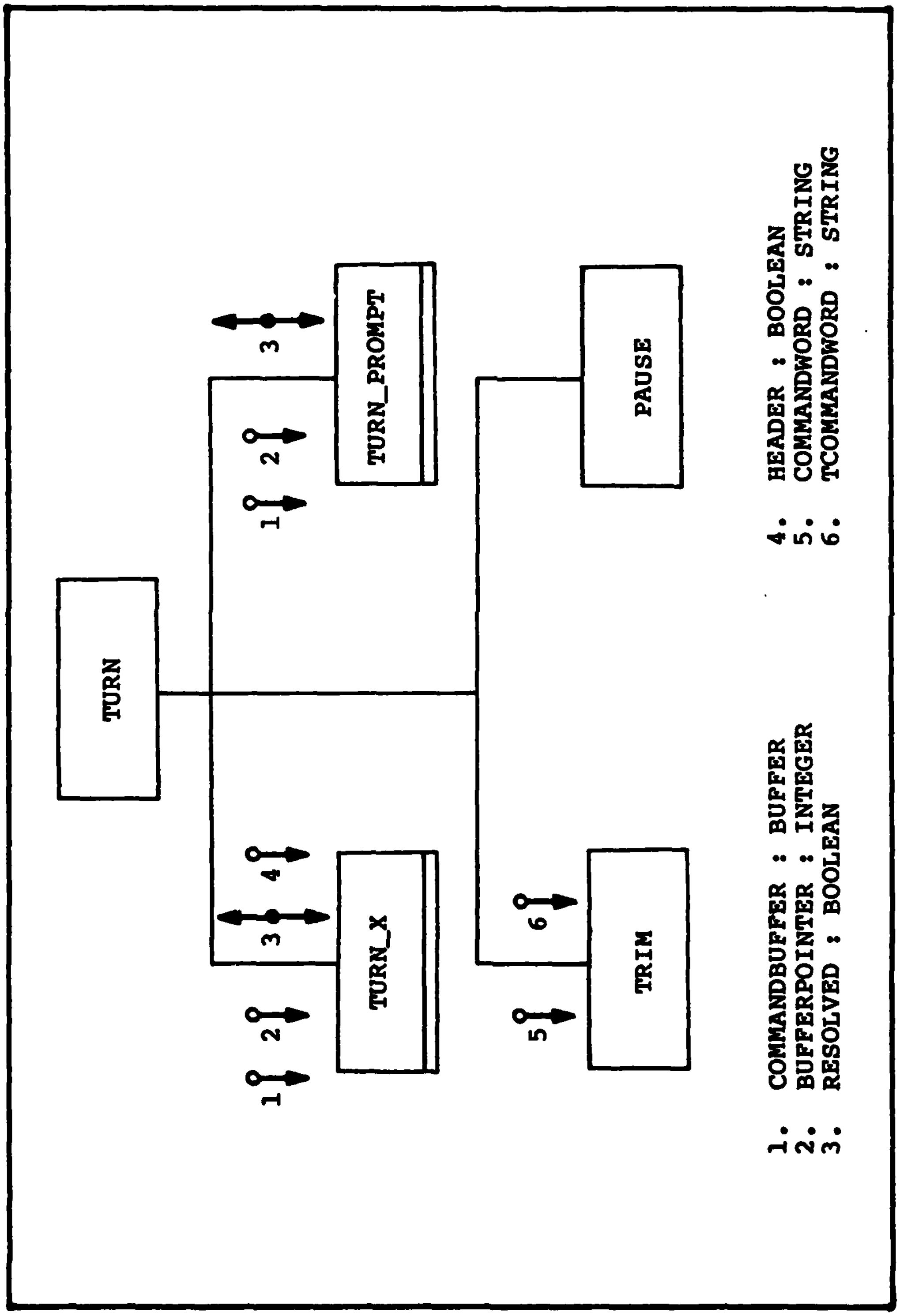


Figure A-24. Structure Chart for Procedure TURN

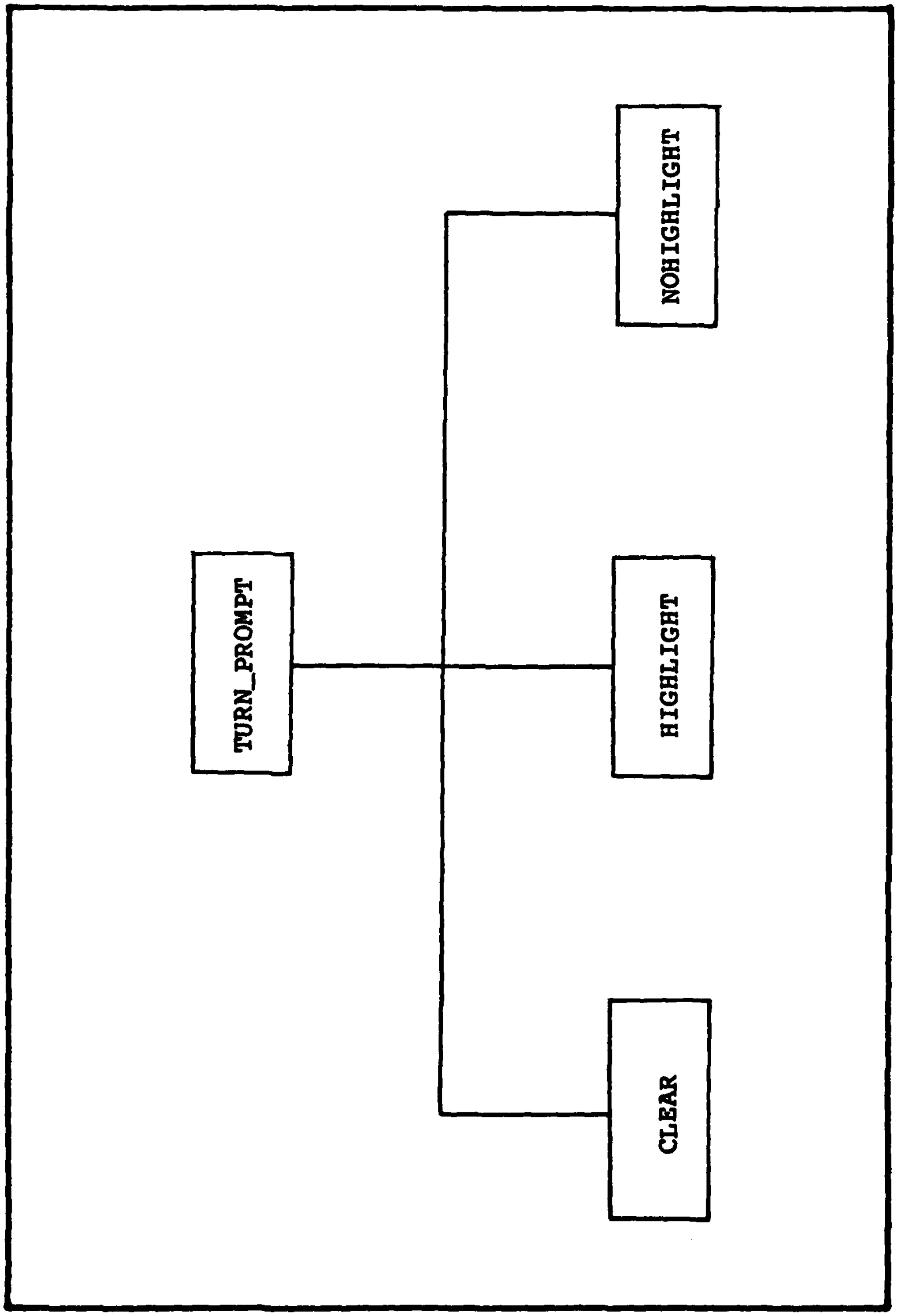


Figure A-25. Structure Chart for Procedure TURN_PROMPT

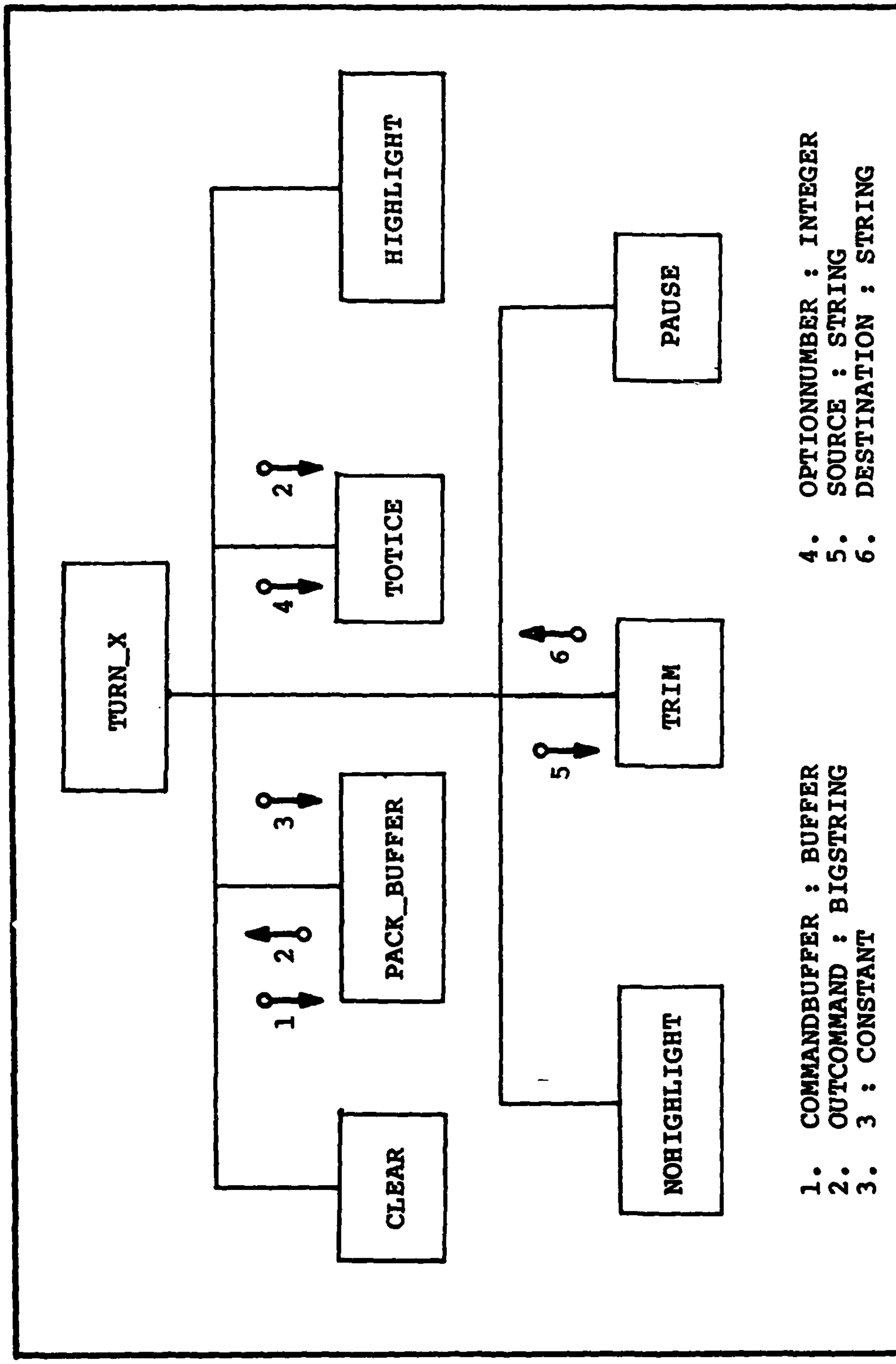


Figure A-26. Structure Chart for Procedure TURN_X

A.5 SUMMARY

The structure charts for the ICECAP modules (main program and subprograms) were included in this appendix in alphabetical order for ease of reference. As new modules are added to the program they can be easily documented in this appendix by including them in alphabetical order and by following the standards that have been defined in this appendix.

APPENDIX B

DATA DICTIONARY

B.1 INTRODUCTION

A data dictionary is data about data. This dictionary documents all of the ICECAP modules, both Pascal and FORTRAN, that have been developed as a result of this thesis effort. Entries for the FORTRAN modules that are contained in VAXTOTAL but used in ICECAP are not included.

A dictionary is not provided for the ICECAP variables. These variables (and constants) are documented in the ICECAP source listing in the form of comments next to the variables in the declaration part of the program.

B.2 DICTIONARY STANDARDS

The entries in this data dictionary conform to the following standards:

- o Entries are in alphabetical order.
- o Entries are for the program, procedures, subroutines, and functions.
- o Data types are shown for the calling parameters and for the variable lists.
- o Categories are listed only when there is information to be provided. If the information is not applicable for a category, the category heading is omitted.
- o Entries are not split between two pages.
- o Each entry is enclosed in its own box.
- o When a module has code unique to the VT100 Terminal, the category "Application" is used.
- o Entries for code written in Pascal are intermingled with entries for code written in FORTRAN.
- o The category "Filename" is used for the FORTRAN modules since each module is separately compiled and kept in a separate file.

B.3 DATA DICTIONARY

```
(****************************************************************)
(*                                                              *)
(*     PROCEDURE:  BOXIT                                        *)
(*                                                              *)
(*     LANGUAGE:  VAX/VMS Pascal                                *)
(*                                                              *)
(*     DESCRIPTION:  Draws a box given the coordinates          *)
(*       of the upper left hand corner of the box and           *)
(*       the width (number of columns) and the depth            *)
(*       (number of rows) of the box.                           *)
(*                                                              *)
(*     APPLICATION:  VT100 Terminal                             *)
(*                                                              *)
(*     CALLING PARAMETERS:                                      *)
(*       ROW : INTEGER                                          *)
(*       COL : INTEGER                                          *)
(*       WIDTH : INTEGER                                        *)
(*       HEIGHT : INTEGER                                       *)
(*                                                              *)
(*     VARIABLES:                                               *)
(*       I : INTEGER                                            *)
(*                                                              *)
(****************************************************************)
```

```
(****************************************************************)
(*                                                              *)
(*     PROCEDURE:  CLEAR                                        *)
(*                                                              *)
(*     LANGUAGE:  VAX/VMS Pascal                                *)
(*                                                              *)
(*     DESCRIPTION:  Clears the screen and places the           *)
(*       cursor in the home position.                           *)
(*                                                              *)
(*     APPLICATION:  VT100 Terminal                             *)
(*                                                              *)
(****************************************************************)
```

DATA DICTIONARY

```
(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  COPY                                    *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Copies a Transfer Function to         *)
(*     another Transfer Function or copies a Matrix to     *)
(*     another Matrix.                                     *)
(*                                                         *)
(*     CALLING PARAMETERS:                                 *)
(*       VAR COMMANDBUFFER : BUFFER                        *)
(*       BUFFERPOINTER : INTEGER                           *)
(*       VAR RESOLVED : BOOLEAN)                           *)
(*                                                         *)
(*     VARIABLES:                                          *)
(*       I : INTEGER                                       *)
(*       MESSAGE : BIGSTRING                               *)
(*       OUTCOMMAND : BIGSTRING                            *)
(*       TCOMMAND1 : STRING                                *)
(*       TCOMMAND2 : STRING                                *)
(*                                                         *)
(***********************************************************)
```

```
(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  CURSORRC                                *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Places cursor at a certain position   *)
(*       on the screen (ROW, COLUMN)                       *)
(*                                                         *)
(*     APPLICATION:  VT100 Terminal                        *)
(*                                                         *)
(*     CALLING PARAMETERS:                                 *)
(*       ROW : INTEGER                                     *)
(*       COL : INTEGER                                     *)
(*                                                         *)
(*     VARIABLES:                                          *)
(*       COL_TENS : CHAR                                   *)
(*       COL_ONES : CHAR                                   *)
(*       ROW_TENS : CHAR                                   *)
(*       ROW_ONES : CHAR                                   *)
(*                                                         *)
(***********************************************************)
```

DATA DICTIONARY

```
(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  DEFINE                                  *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Looks for a legal object of the       *)
(*       command word DEFINE and takes appropriate         *)
(*       action.                                           *)
(*                                                         *)
(*     CALLING PARAMETERS:                                 *)
(*       VAR COMMANDBUFFER : BUFFER                        *)
(*       VAR BUFFERPOINTER : INTEGER                       *)
(*       VAR RESOLVED : BOOLEAN                            *)
(*                                                         *)
(*     VARIABLES:                                          *)
(*       MATCH : BOOLEAN                                   *)
(*       OPTIONNUMBER : INTEGER                            *)
(*                                                         *)
(***********************************************************)
```

```
(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  DEFINE_PROMPT                           *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Provides information on the legal     *)
(*       object of the command word DEFINE and waits for   *)
(*       user response.                                    *)
(*                                                         *)
(*     CALLING PARAMETERS:                                 *)
(*       VAR COMMANDBUFFER : BUFFER                        *)
(*       VAR BUFFERPOINTER : INTEGER                       *)
(*       VAR RESOLVED : BOOLEAN)                           *)
(*                                                         *)
(***********************************************************)
```

```
(*****************************************************************)
(*                                                               *)
(*     PROCEDURE:  DEFINE_TF                                     *)
(*                                                               *)
(*     LANGUAGE:  VAX/VMS Pascal                                 *)
(*                                                               *)
(*     DESCRIPTION:  Looks for a legal object of the             *)
(*       command string DEFINE (transfer function)               *)
(*       and takes appropriate action.                           *)
(*                                                               *)
(*     CALLING PARAMETERS:                                       *)
(*       VAR COMMANDBUFFER : BUFFER                              *)
(*       BUFFERPOINTER : INTEGER                                 *)
(*       VAR RESOLVED : BOOLEAN                                  *)
(*                                                               *)
(*     VARIABLES:                                                *)
(*       TCOMMANDWORD : STRING                                   *)
(*                                                               *)
(*****************************************************************)
```

```
(*****************************************************************)
(*                                                               *)
(*     PROCEDURE:  DEF_TF_PLANE                                  *)
(*                                                               *)
(*     LANGUAGE:  VAX/VMS Pascal                                 *)
(*                                                               *)
(*     DESCRIPTION:  Processes the object of DEFINE              *)
(*     (transfer function) (POLY or FACT) to determine           *)
(*     which PLANE is desired.                                   *)
(*                                                               *)
(*     CALLING PARAMETERS:                                       *)
(*       VAR COMMANDBUFFER : BUFFER                              *)
(*       BUFFERPOINTER : INTEGER                                 *)
(*       VAR RESOLVED : BOOLEAN                                  *)
(*                                                               *)
(*     VARIABLES:                                                *)
(*       COMMANDWORD : STRING                                    *)
(*       OPTIONNUMBER : INTEGER                                  *)
(*                                                               *)
(*****************************************************************)
```

```
(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  DICTIONARY                              *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Checks each word in a command string *)
(*       against all legal ICECAP KEYWORDs.  It notifies   *)
(*       the user of all illegal command words used in     *)
(*       the command string.  If it finds any illegal      *)
(*       command words it returns ALLMATCH = FALSE.        *)
(*                                                         *)
(*     CALLING PARAMETERS:                                 *)
(*       VAR ALLMATCH : BOOLEAN                            *)
(*       VAR COMMANDBUFFER : BUFFER                        *)
(*       BUFFERPOINTER : INTEGER                           *)
(*                                                         *)
(*     VARIABLES:                                          *)
(*       MATCH : BOOLEAN                                   *)
(*                                                         *)
(***********************************************************)


(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  DISPLAY_OR_PRIN                         *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Displays item on screen or prints     *)
(*     item in ANSWER File.                                *)
(*                                                         *)
(*     CALLING PARAMETERS:                                 *)
(*       COMMANDBUFFER : BUFFER                            *)
(*       BUFFERPOINTER : INTEGER                           *)
(*       VAR RESOLVED : BOOLEAN                            *)
(*                                                         *)
(*     VARIABLES:                                          *)
(*       ANSWERFLAG : BOOLEAN                              *)
(*       I : INTEGER                                       *)
(*       MESSAGE : BIGSTRING                               *)
(*       MESSAGEBUFFER : BUFFER                            *)
(*       OUTCOMMAND : BIGSTRING                            *)
(*       TCOMMAND1 : STRING                                *)
(*       TCOMMAND2 : STRING                                *)
(*       WHERE : STRING                                    *)
(*                                                         *)
(***********************************************************)
```

DATA DICTIONARY

```
(***********************************************************)
(*                                                         *)
(*     SUBROUTINE:  FIND_BORDERS                           *)
(*                                                         *)
(*     FILENAME:  FINDBORD.ICE                             *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS FORTRAN                          *)
(*                                                         *)
(*     DESCRIPTION:  Finds the four borders for the Root   *)
(*       Locus plot (known in TOTAL as AA, BB, CC, DD)     *)
(*       based on the poles and zeros of OLTF              *)
(*                                                         *)
(*     VARIABLES:                                          *)
(*       AA : REAL*4                                       *)
(*       BB : REAL*4                                       *)
(*       CC : REAL*4                                       *)
(*       DD : REAL*4                                       *)
(*       DEBUG : LOGICAL                                   *)
(*       EAST : INTEGER                                    *)
(*       HEIGHT : REAL*4                                   *)
(*       I : INTEGER                                       *)
(*       LENGTH : REAL*4                                   *)
(*       NOLP : INTEGER                                    *)
(*       NOLZ : INTEGER                                    *)
(*       OLPOLE : MATRIX [50,2] OF REAL*4                  *)
(*       OLZERO : MATRIX [50,2] OF REAL*4                  *)
(*       SOUTH : INTEGER                                   *)
(*       WEST : INTEGER                                    *)
(*                                                         *)
(***********************************************************)
```

```
(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  FORM                                    *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Forms OLTF or CLTF depending upon     *)
(*     user's choice.                                      *)
(*                                                         *)
(*     CALLING PARAMETERS:                                 *)
(*       VAR COMMANDBUFFER : BUFFER                        *)
(*       BUFFERPOINTER : INTEGER                           *)
(*       VAR RESOLVED : BOOLEAN                            *)
(*                                                         *)
(***********************************************************)
```

DATA DICTIONARY

```
(*****************************************************************)
(*                                                               *)
(*     PROCEDURE:  GRAPHICS                                      *)
(*                                                               *)
(*     LANGUAGE:  VAX/VMS Pascal                                 *)
(*                                                               *)
(*     DESCRIPTION:  Places the terminal in the graphics         *)
(*       mode so that the Special Graphics Characters in         *)
(*       Table 3-9 of the VT100 User Guide can be used.          *)
(*                                                               *)
(*     APPLICATION:  VT100 Terminal                              *)
(*                                                               *)
(*****************************************************************)


(*****************************************************************)
(*                                                               *)
(*     PROCEDURE:  HELP                                          *)
(*                                                               *)
(*     LANGUAGE:  VAX/VMS Pascal                                 *)
(*                                                               *)
(*     DESCRIPTION:  Looks for a legal object of the             *)
(*       command word HELP and takes appropriate                 *)
(*       action.                                                 *)
(*                                                               *)
(*     CALLING PARAMETERS:                                       *)
(*       VAR COMMANDBUFFER : BUFFER                              *)
(*       VAR BUFFERPOINTER : INTEGER                             *)
(*       VAR RESOLVED : BOOLEAN                                  *)
(*                                                               *)
(*     VARIABLES:                                                *)
(*       COMMANDWORD : STRING                                    *)
(*       TCOMMANDWORD : STRING                                   *)
(*                                                               *)
(*****************************************************************)


(*****************************************************************)
(*                                                               *)
(*     PROCEDURE:  HELP_COPY                                     *)
(*                                                               *)
(*     LANGUAGE:  VAX/VMS Pascal                                 *)
(*                                                               *)
(*     DESCRIPTION:  Explains how to use the COPY                *)
(*       command.                                                *)
(*                                                               *)
(*     APPLICATION:  VT100 Terminal                              *)
(*                                                               *)
(*****************************************************************)
```

DATA DICTIONARY

```
(****************************************************************)
(*                                                              *)
(*     PROCEDURE:  HELP_INITIAL                                 *)
(*                                                              *)
(*     LANGUAGE:  VAX/VMS Pascal                                *)
(*                                                              *)
(*     DESCRIPTION:  Displays all valid Command Words           *)
(*     that can be used to start a Command String.              *)
(*                                                              *)
(****************************************************************)
```

```
(****************************************************************)
(*                                                              *)
(*     PROCEDURE:  HELP_PROMPT                                  *)
(*                                                              *)
(*     LANGUAGE:  VAX/VMS Pascal                                *)
(*                                                              *)
(*     DESCRIPTION:  Displays all valid Command Words           *)
(*     that can be used to start a Command String.              *)
(*                                                              *)
(****************************************************************)
```

```
(****************************************************************)
(*                                                              *)
(*     PROCEDURE:  HELP_SYSTEM                                  *)
(*                                                              *)
(*     LANGUAGE:  VAX/VMS Pascal                                *)
(*                                                              *)
(*     DESCRIPTION:  Displays all valid ICECAP command          *)
(*     words in alphabetical order.                             *)
(*                                                              *)
(****************************************************************)
```

```
(****************************************************************)
(*                                                              *)
(*     PROCEDURE:  HIGHLIGHT                                    *)
(*                                                              *)
(*     LANGUAGE:  VAX/VMS Pascal                                *)
(*                                                              *)
(*     DESCRIPTION:  Puts all subsequent characters into        *)
(*     reverse video until NOHIGHLIGHT is called.               *)
(*                                                              *)
(*     APPLICATION:  VT100 Terminal                             *)
(*                                                              *)
(****************************************************************)
```

DATA DICTIONARY

```
(**************************************************************)
(*                                                            *)
(*     PROGRAM:  ICER                                         *)
(*                                                            *)
(*     FILENAME:  ICER.PAS                                    *)
(*                                                            *)
(*     LANGUAGE:  VAX/VMS Pascal and FORTRAN                  *)
(*                                                            *)
(*     DESCRIPTION:  ICECAP - Interactive Control             *)
(*        Engineering Computer Analysis Package               *)
(*                                                            *)
(*     APPLICATION:  VT100 Terminal                           *)
(*                                                            *)
(*     DATE OF REVISION:  16 AUG 82                           *)
(*                                                            *)
(*     AUTHOR:  Major Charles J. Gembarowski                  *)
(*                                                            *)
(*     CONSTANTS:                                             *)
(*       BUFFERSIZE = 10                                      *)
(*       COMMANDSIZE = 80                                     *)
(*       WORDSIZE = 12                                        *)
(*                                                            *)
(*     TYPES:                                                 *)
(*       BIGSTRING = PACKED ARRAY[1..COMMANDSIZE] OF CHAR *)
(*       STRING = PACKED ARRAY[1..WORDSIZE] OF CHAR           *)
(*       BUFFER = ARRAY[1..BUFFERSIZE] OF STRING              *)
(*                                                            *)
(*     VARIABLES:                                             *)
(*       BUFFERPOINTER : INTEGER                              *)
(*       CLC : INTEGER                                        *)
(*       COMMAND : ARRAY[1..COMMANDSIZE] OF CHAR              *)
(*       COMMANDBUFFER : BUFFER                               *)
(*       COMMANDWORD : STRING                                 *)
(*       CONTINUE : CHAR                                      *)
(*       HEADER : BOOLEAN                                     *)
(*       I : INTEGER                                          *)
(*       ICOMMAND : BIGSTRING                                 *)
(*       KEYWORD : STRING                                     *)
(*       LETTER : CHAR                                        *)
(*       LINE : BIGSTRING                                     *)
(*       OPTIONNUMBER : INTEGER                               *)
(*       PRINTFLAG : BOOLEAN                                  *)
(*       RESOLVED : BOOLEAN                                   *)
(*       STATUS : INTEGER                                     *)
(*       TCOMMANDWORD : STRING                                *)
(*       UCOMMAND : BIGSTRING                                 *)
(*       WLC : INTEGER                                        *)
(*       WORD : ARRAY[1..WORDSIZE] OF CHAR                    *)
(*                                                            *)
(**************************************************************)
```

```
(*************************************************************)
(*                                                           *)
(*     PROCEDURE:  INTERPRET                                 *)
(*                                                           *)
(*     LANGUAGE:  VAX/VMS Pascal                             *)
(*                                                           *)
(*     DESCRIPTION:  Reads command words out of the          *)
(*       command buffer and calls appropriate procedures     *)
(*       for action.                                         *)
(*                                                           *)
(*     CALLING PARAMETERS:                                   *)
(*       VAR COMMANDBUFFER : BUFFER                          *)
(*       VAR BUFFERPOINTER : INTEGER                         *)
(*       VAR RESOLVED : BOOLEAN                              *)
(*       VAR PRINTFLAG : BOOLEAN                             *)
(*                                                           *)
(*     VARIABLES:                                            *)
(*       ALLMATCH : BOOLEAN                                  *)
(*       COMMANDWORD : STRING                                *)
(*       OUTCOMMAND : BIGSTRING                              *)
(*       TCOMMANDWORD : STRING                               *)
(*       OPTIONNUMBER : INTEGER                              *)
(*                                                           *)
(*************************************************************)
```

```
(*************************************************************)
(*                                                           *)
(*     PROCEDURE:  LOCUS                                     *)
(*                                                           *)
(*     LANGUAGE:  VAX/VMS Pascal                             *)
(*                                                           *)
(*     DESCRIPTION:  Displays, prints or plot the Root       *)
(*     Locus for the OLTF which must be already defined.     *)
(*                                                           *)
(*     APPLICATION:  VT100 Terminal                          *)
(*                                                           *)
(*     CALLING PARAMETERS:                                   *)
(*       VAR COMMANDBUFFER : BUFFER                          *)
(*       BUFFERPOINTER : INTEGER                             *)
(*       VAR RESOLVED : BOOLEAN                              *)
(*                                                           *)
(*************************************************************)
```

```
(***************************************************************)
(*                                                             *)
(*     PROCEDURE:  LOCUS_AUTOSCALE                             *)
(*                                                             *)
(*     LANGUAGE:  VAX/VMS Pascal                               *)
(*                                                             *)
(*     DESCRIPTION:  Displays, prints or plot the Root         *)
(*     Locus for the OLTF which must be already defined.       *)
(*     Chooses the borders based on the locations of           *)
(*     poles and zeroes.                                       *)
(*                                                             *)
(*     APPLICATION:  VT100 Terminal                            *)
(*                                                             *)
(***************************************************************)

(***************************************************************)
(*                                                             *)
(*     PROCEDURE:  LOCUS_MAGNIFY                               *)
(*                                                             *)
(*     LANGUAGE:  VAX/VMS Pascal                               *)
(*                                                             *)
(*     DESCRIPTION:  Displays, prints or plot the Root         *)
(*     Locus for the OLTF which must be already defined.       *)
(*     Doubles the size of the locus from the last time        *)
(*     it was shown.                                           *)
(*                                                             *)
(*     APPLICATION:  VT100 Terminal                            *)
(*                                                             *)
(***************************************************************)

(***************************************************************)
(*                                                             *)
(*     PROCEDURE:  LOCUS_SHRINK                                *)
(*                                                             *)
(*     LANGUAGE:  VAX/VMS Pascal                               *)
(*                                                             *)
(*     DESCRIPTION:  Displays, prints or plot the Root         *)
(*     Locus for the OLTF which must be already defined.       *)
(*     Shrinks the size of the locus by a factor of 2.         *)
(*                                                             *)
(*     APPLICATION:  VT100 Terminal                            *)
(*                                                             *)
(***************************************************************)
```

DATA DICTIONARY

```
(**********************************************************)
(*                                                        *)
(*     PROCEDURE:  LOCUS_ZOOM                             *)
(*                                                        *)
(*     LANGUAGE:  VAX/VMS Pascal                          *)
(*                                                        *)
(*     DESCRIPTION:  Displays, prints or plot the Root    *)
(*     Locus for the OLTF which must be already defined.  *)
(*     User chooses center point and the horizontal       *)
(*     distance to the border                             *)
(*                                                        *)
(*     APPLICATION:  VT100 Terminal                       *)
(*                                                        *)
(**********************************************************)
```

```
(**********************************************************)
(*                                                        *)
(*     SUBROUTINE:  MAGNIFY                               *)
(*                                                        *)
(*     FILENAME:  MAGNIFY.ICE                             *)
(*                                                        *)
(*     LANGUAGE:  VAX/VMS FORTRAN                         *)
(*                                                        *)
(*     DESCRIPTION:  Magnifies the Root Locus by finding  *)
(*       new borders for the Root Locus plot.  Does this  *)
(*       by dividing the present values of AA, BB, CC,    *)
(*       and DD by two.                                   *)
(*                                                        *)
(*     VARIABLES:                                         *)
(*       AA : REAL*4                                      *)
(*       BB : REAL*4                                      *)
(*       CC : REAL*4                                      *)
(*       DD : REAL*4                                      *)
(*                                                        *)
(**********************************************************)
```

DATA DICTIONARY

```
(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  NOGRAPHICS                              *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Takes the VT100 terminal out of the   *)
(*       graphics mode.  Restores the lowercase character  *)
(*       set.                                              *)
(*                                                         *)
(*     APPLICATION:  VT100 Terminal                        *)
(*                                                         *)
(***********************************************************)


(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  NOHIGHLIGHT                             *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Puts all subsequent characters into   *)
(*     normal video.                                       *)
(*                                                         *)
(*     APPLICATION:  VT100 Terminal                        *)
(*                                                         *)
(***********************************************************)


(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  PACK_BUFFER                             *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Takes the COMMANDBUFFER and packs     *)
(*       it into OUTCOMMAND which will go to TOTAL.        *)
(*                                                         *)
(*     CALLING PARAMETERS:                                 *)
(*       COMMANDBUFFER : BUFFER                            *)
(*       VAR OUTCOMMAND : BIGSTRING                        *)
(*       COMMANDLEVEL : INTEGER                            *)
(*                                                         *)
(*     VARIABLES:                                          *)
(*       I : INTEGER                                       *)
(*       J : INTEGER                                       *)
(*       OUTARRAY : ARRAY[1..COMMANDSIZE] OF CHAR          *)
(*       TEMP : ARRAY[1..WORDSIZE] OF CHAR                 *)
(***********************************************************)
```

DATA DICTIONARY

```
(*************************************************************)
(*                                                           *)
(*     PROCEDURE:  PAUSE                                     *)
(*                                                           *)
(*     LANGUAGE:  VAX/VMS Pascal                             *)
(*                                                           *)
(*     DESCRIPTION:  Causes program to halt until            *)
(*       <RETURN> is pressed, thereby allowing the user      *)
(*       time to read the screen.                            *)
(*                                                           *)
(*     APPLICATION:  VT100 Terminal                          *)
(*                                                           *)
(*     VARIABLES:                                            *)
(*       CONTINUE : CHAR                                     *)
(*                                                           *)
(*************************************************************)
```

```
(*************************************************************)
(*                                                           *)
(*     PROCEDURE:  PRINT_BUFFER                              *)
(*                                                           *)
(*     LANGUAGE:  VAX/VMS Pascal                             *)
(*                                                           *)
(*     DESCRIPTION:  Prints out entire command buffer        *)
(*     with no leading blanks, one trailing blank, no        *)
(*     abbreviations, and with one space between words.      *)
(*     Words are in upppercase.                              *)
(*                                                           *)
(*     CALLING PARAMETERS:                                   *)
(*       COMMANDBUFFER: BUFFER                               *)
(*       BUFFERPOINTER : INTEGER                             *)
(*                                                           *)
(*     VARIABLES:                                            *)
(*       TCOMMANDWORD : STRING                               *)
(*                                                           *)
(*************************************************************)
```

```
(***************************************************************)
(*                                                             *)
(*     PROCEDURE:  READCOM                                     *)
(*                                                             *)
(*     LANGUAGE:  VAX/VMS Pascal                               *)
(*                                                             *)
(*     DESCRIPTION:  Reads in ICOMMAND (until <CR>),           *)
(*       changes it to uppercase (UCOMMAND), breaks it         *)
(*       into command words, and puts the command words        *)
(*       into COMMANDBUFFER.                                   *)
(*                                                             *)
(*     CALLING PARAMETERS:                                     *)
(*       VAR COMMANDBUFFER : BUFFER                            *)
(*       VAR BUFFERPOINTER : INTEGER                           *)
(*                                                             *)
(*     VARIABLES:                                              *)
(*       COMMANDWORD : STRING                                  *)
(*       TCOMMANDWORD : STRING                                 *)
(*                                                             *)
(***************************************************************)
```

```
(***************************************************************)
(*                                                             *)
(*     FUNCTION:  STR$UPCASE                                   *)
(*                                                             *)
(*     DESCRIPTION:  Puts all letters into uppercase           *)
(*                                                             *)
(*     CALLING PARAMETERS:                                     *)
(*       %STDESCR UCOMMAND : BIGSTRING                         *)
(*       %STDESCR ICOMMAND : BIGSTRING                         *)
(*                                                             *)
(***************************************************************)
```

DATA DICTIONARY

```
(***********************************************************)
(*                                                         *)
(*     SUBROUTINE:  SHRINK                                 *)
(*                                                         *)
(*     FILENAME:  SHRINK.ICE                               *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS FORTRAN                          *)
(*                                                         *)
(*     DESCRIPTION:  Shrinks the Root Locus by finding     *)
(*       new borders for the Root Locus plot.  Does this   *)
(*       by multiplying the present values of AA, BB,      *)
(*       CC, and DD by two.                                *)
(*                                                         *)
(*     APPLICATION:  VT100 Terminal                        *)
(*                                                         *)
(*     VARIABLES:                                          *)
(*       AA : REAL*4                                       *)
(*       BB : REAL*4                                       *)
(*       CC : REAL*4                                       *)
(*       DD : REAL*4                                       *)
(*                                                         *)
(***********************************************************)
```

```
(***********************************************************)
(*                                                         *)
(*     PROCEDURE:  TITLE_SLIDE                             *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS Pascal                           *)
(*                                                         *)
(*     DESCRIPTION:  Displays initial screen showing       *)
(*     ICECAP in large letters and copyright information.  *)
(*                                                         *)
(*     APPLICATION:  VT100 Terminal                        *)
(*                                                         *)
(***********************************************************)
```

DATA DICTIONARY

```
(*************************************************************)
(*                                                           *)
(*     SUBROUTINE:  TOTICE                                   *)
(*                                                           *)
(*     FILENAME:  TOTICE.ICE                                 *)
(*                                                           *)
(*     LANGUAGE:  VAX/VMS FORTRAN                            *)
(*                                                           *)
(*     DESCRIPTION:  Interfaces the Pascal portion of        *)
(*       ICECAP with the FORTRAN portion by passing          *)
(*       the option numbers and commands sent to it by       *)
(*       the Pascal portion to the FORTRAN portion.          *)
(*                                                           *)
(*     CALLING PARAMETERS:                                   *)
(*       OPTIONNUMBER : INTEGER                              *)
(*       %STDESCR LINE : PACKED ARRAY[INTEGER] OF CHAR       *)
(*                                                           *)
(*     VARIABLES:                                            *)
(*       DATM : MATRIX [100] OF REAL*4                       *)
(*       DEBUG : LOGICAL                                     *)
(*       GOPLOT : LOGICAL                                    *)
(*       I : INTEGER                                         *)
(*       IPLOT : INTEGER                                     *)
(*       JFLAG : MATRIX [100] OF INTEGER                     *)
(*       LASTOPT : INTEGER                                   *)
(*       LINE : STRING [80] OF CHAR                          *)
(*       LOPT : INTEGER                                      *)
(*       MCOMM : MATRIX [100] OF INTEGER                     *)
(*       MPT : INTEGER                                       *)
(*       NNU : INTEGER                                       *)
(*       NOPT : INTEGER                                      *)
(*       NROUTE : MATRIX [10] OF INTEGER                     *)
(*       NRPT : INTEGER                                      *)
(*       OPTN : INTEGER                                      *)
(*       REQUEST : LOGICAL                                   *)
(*                                                           *)
(*************************************************************)
```

DATA DICTIONARY

```
(***********************************************************)
(*                                                         *)
(*     SUBROUTINE:  TOTINI                                 *)
(*                                                         *)
(*     FILENAME:  TOTINI.ICE                               *)
(*                                                         *)
(*     LANGUAGE:  VAX/VMS FORTRAN                          *)
(*                                                         *)
(*     DESCRIPTION:  Initializes the FORTRAN modules in    *)
(*       ICECAP as part of the ICECAP initialization       *)
(*       process.                                          *)
(*                                                         *)
(*     VARIABLES:                                          *)
(*       ALREADY : LOGICAL                                 *)
(*       ANSWER : LOGICAL                                  *)
(*       CALC : LOGICAL                                    *)
(*       CLNPOLY : MATRIX [51] OF REAL*4                   *)
(*       CLOSED : LOGICAL                                  *)
(*       DEBUG : LOGICAL                                   *)
(*       DECIBEL : LOGICAL                                 *)
(*       DEGREE : LOGICAL                                  *)
(*       DMAT : MATRIX [10,10] OF REAL*4                   *)
(*       ECHO : LOGICAL                                    *)
(*       EXTCALC : LOGICAL                                 *)
(*       FILOPN : LOGICAL                                  *)
(*       GRID : LOGICAL                                    *)
(*       HERTZ : LOGICAL                                   *)
(*       INMASS : MATRIX [47] OF INTEGER                   *)
(*       KFLAG : MATRIX [20] OF INTEGER                    *)
(*       LFLAGE : MATRIX [20] OF INTEGER                   *)
(*       NCALL : MATRIX [20] OF INTEGER                    *)
(*       NGO : INTEGER                                     *)
(*       PLAT : LOGICAL                                    *)
(*       REQUEST : LOGICAL                                 *)
(*       SCALE : LOGICAL                                   *)
(*       TEKPLOT : LOGICAL                                 *)
(*       TEST : LOGICAL                                    *)
(*                                                         *)
(***********************************************************)
```

DATA DICTIONARY

```
(*************************************************************)
(*                                                           *)
(*     PROCEDURE:  TRIM                                      *)
(*                                                           *)
(*     LANGUAGE:  VAX/VMS Pascal                             *)
(*                                                           *)
(*     DESCRIPTION:  Trims the trailing blanks off of        *)
(*       the SOURCE string and places the stripped           *)
(*       version into the DESTINATION string.                *)
(*                                                           *)
(*     CALLING PARAMETERS:                                   *)
(*       VAR SOURCE : STRING                                 *)
(*       VAR DESTINATION : STRING                            *)
(*                                                           *)
(*     VARIABLES:                                            *)
(*       I : INTEGER                                         *)
(*       UDESTINATION : ARRAY[1..WORDSIZE] OF CHAR           *)
(*       USOURCE : ARRAY[1..WORDSIZE] OF CHAR                *)
(*                                                           *)
(*************************************************************)
```

```
(*************************************************************)
(*                                                           *)
(*     PROCEDURE:  TURN                                      *)
(*                                                           *)
(*     LANGUAGE:  VAX/VMS Pascal                             *)
(*                                                           *)
(*     DESCRIPTION:  Used to turn the various control        *)
(*     switches ON and OFF.                                  *)
(*                                                           *)
(*     CALLING PARAMETERS:                                   *)
(*       VAR COMMANDBUFFER : BUFFER                          *)
(*       VAR BUFFERPOINTER : INTEGER                         *)
(*       VAR RESOLVED : BOOLEAN                              *)
(*                                                           *)
(*     VARIABLES:                                            *)
(*       COMMANDWORD : STRING                                *)
(*       TCOMMANDWORD : STRING                               *)
(*                                                           *)
(*************************************************************)
```

DATA DICTIONARY

```
(****************************************************************)
(*                                                              *)
(*     PROCEDURE:  TURN_PROMPT                                  *)
(*                                                              *)
(*     LANGUAGE:  VAX/VMS Pascal                                *)
(*                                                              *)
(*     DESCRIPTION:  In the absence of an object for            *)
(*     TURN, it prompts for one.                                *)
(*                                                              *)
(*     CALLING PARAMETERS:                                      *)
(*       VAR COMMANDBUFFER : BUFFER                             *)
(*       VAR BUFFERPOINTER : INTEGER                            *)
(*       VAR RESOLVED : BOOLEAN                                 *)
(*                                                              *)
(****************************************************************)


(****************************************************************)
(*                                                              *)
(*     PROCEDURE:  TURN_X                                       *)
(*                                                              *)
(*     LANGUAGE:  VAX/VMS Pascal                                *)
(*                                                              *)
(*     DESCRIPTION:  Processes the object of TURN by            *)
(*     looking for OFF or ON and sets the switch                *)
(*     accordingly.                                             *)
(*                                                              *)
(*     CALLING PARAMETERS:                                      *)
(*       VAR COMMANDBUFFER : BUFFER                             *)
(*       VAR BUFFERPOINTER : INTEGER                            *)
(*       VAR RESOLVED : BOOLEAN                                 *)
(*       VAR HEADER : BOOLEAN                                   *)
(*                                                              *)
(*     VARIABLES:                                               *)
(*       OPTIONNUMBER : INTEGER                                 *)
(*       OUTCOMMAND : BIGSTRING                                 *)
(*       TCOMMANDWORD : STRING                                  *)
(*                                                              *)
(****************************************************************)
```

B.4 DATA DICTIONARY BLANK FORM

A blank data dictionary entry form is included below and may be reproduced as necessary for adding new items to the data dictionary.

```
(**********************************************************)
(*                                                        *)
(*     PROCEDURE: ___________________________________     *)
(*                                                        *)
(*     LANGUAGE:  VAX/VMS ____________________________    *)
(*                                                        *)
(*     DESCRIPTION:  ________________________________     *)
(*                                                        *)
(*     ______________________________________________     *)
(*                                                        *)
(*     ______________________________________________     *)
(*                                                        *)
(*     ______________________________________________     *)
(*                                                        *)
(*     APPLICATION:  VT100 Terminal                       *)
(*                                                        *)
(*     CALLING PARAMETERS:                                *)
(*                                                        *)
(*       ____________________________________________     *)
(*                                                        *)
(*       ____________________________________________     *)
(*                                                        *)
(*       ____________________________________________     *)
(*                                                        *)
(*     VARIABLES:                                         *)
(*                                                        *)
(*       ____________________________________________     *)
(*                                                        *)
(*       ____________________________________________     *)
(*                                                        *)
(*       ____________________________________________     *)
(*                                                        *)
(*       ____________________________________________     *)
(*                                                        *)
(*       ____________________________________________     *)
(*                                                        *)
(*       ____________________________________________     *)
(*                                                        *)
(**********************************************************)
```

B.5 SUMMARY

The data dictionary for the program, procedures, functions, and subroutines of ICECAP has been provided along with a blank dictionary entry form for use in further documenting the ICECAP as it develops through subsequent thesis efforts. Dictionary type information on the ICECAP variables can be found in the ICECAP source listing.

APPENDIX C

PROBLEM REPORTS

C.1 INTRODUCTION

This appendix documents the problem reports generated as a result of having analyzed and executed TOTAL, VAXTOTAL, and ICECAP. These problem reports can serve as a basis for future development of ICECAP. A blank problem report form has been included. This form can be reproduced as necessary so that a continuous record of program problems and their corrections can be maintained.

C.2 PROBLEM REPORTS

The following pages contain the problem reports that have been accumulated to date. There is one report per page.

PROBLEM REPORTS

PROBLEM REPORT NUMBER: 1

DATE: 1 JUL 82

ORIGINATOR: Major Charles J. Gembarowski

PROBLEM NAME: DEL

PROGRAM(S) HAVING PROBLEM: TOTAL, VAXTOTAL, ICECAP

PROBLEM DESCRIPTION: Option 49 resets DEL and DELPR to zero after printing out their value. Cause is Subroutine ROOT10, lines 0181 and 0182. Result is that the Root Locus Options bomb out.

PROBLEM SERIOUSNESS: Very

DIFFICULTY OF FIX: Easy

SUGGESTIONS FOR FIX: Avoid that section of code by inserting the following line of code: "IF NOPT N.EQ.49 THEN".

DISPOSITION: Fix is implemented and has been tested in ICECAP.

PROBLEM REPORTS

PROBLEM REPORT NUMBER: 2

DATE: 23 JUN 82

ORIGINATOR: Major Charles J. Gembarowski

PROBLEM NAME: Output

PROGRAM(S) HAVING PROBLEM: TOTAL, VAXTOTAL, ICECAP

PROBLEM DESCRIPTION: When writing output to the ANSWER.DAT file, the user does not see what is being written.

PROBLEM SERIOUSNESS: Major inconvenience

DIFFICULTY OF FIX: Medium

SUGGESTIONS FOR FIX: Write the same information to the screen and to the file in sequence so that user can see what is being written.

DISPOSITION: This technique has been implemented and tested in ICECAP.

PROBLEM REPORTS

PROBLEM REPORT NUMBER: 3

DATE: 3 JUN 82

ORIGINATOR: Major Charles J. Gembarowski

PROBLEM NAME: Root Locus Scaling

PROGRAM(S) HAVING PROBLEM: TOTAL, VAXTOTAL, ICECAP

PROBLEM DESCRIPTION: User should not have to figure out the values to assign to AA, BB, CC, and DD in order to establish reasonable scales for a Root Locus plot. Default values are not adequate.

PROBLEM SERIOUSNESS: Major inconvenience

DIFFICULTY OF FIX: Medium

SUGGESTIONS FOR FIX: Implement autoscaling based on the locations of the poles and zeroes of the Open Loop Transfer Function. Show only the top half of the Root Locus (with only a small portion of the bottom half for context) since the bottom half of the Root Locus is a mirror image of the top half. Allow the user to magnify and shrink the Root Locus as desired as well as zoom about a chosen point.

DISPOSITION: Autoscaling, magnification, and shrinking has been implemented and tested in ICECAP. The structure and command language of ICECAP is ready for the implementation of the zoom feature.

PROBLEM REPORTS

PROBLEM REPORT NUMBER: 4

DATE: 18 JUL 82

ORIGINATOR: Major Charles J. Gembarowski

PROBLEM NAME: Precision

PROGRAM(S) HAVING PROBLEM: VAXTOTAL, ICECAP

PROBLEM DESCRIPTION: The fact that the VAX is less precise than the Cyber could cause calculation errors, such as divide by zero, and wrong decisions.

PROBLEM SERIOUSNESS: Unknown

DIFFICULTY OF FIX: Massive

SUGGESTIONS FOR FIX: Fix occurrences as they happen and as they are understood. Methodically replace some of the cruder algorithms of TOTAL with more numerically precise algorithms.

DISPOSITION: Continuous effort. However, PI, the variable representing the transcendental number pi, has been made more precise in ICECAP to make the angle calculations more reliable particularly in the root locus options.

PROBLEM REPORTS

PROBLEM REPORT NUMBER: 5

DATE: 25 JUL 82

ORIGINATOR: Major Charles J. Gembarowski

PROBLEM NAME: Root-finder

PROGRAM(S) HAVING PROBLEM: TOTAL, VAXTOTAL, ICECAP

PROBLEM DESCRIPTION: The programs are oftentimes unable to find the roots of polynomials properly, particularly when there are multiple roots involved. Several different root-finders are implemented in the programs but the user does not know when and why to select from among them.

PROBLEM SERIOUSNESS: Very

DIFFICULTY OF FIX: Medium

SUGGESTIONS FOR FIX: Strip out all of the present root-finders and use one good one.

DISPOSITION: All of the root-finders have been stripped out of ICECAP and replaced with a library routine ZRPOLY that uses the Jenkins-Traub method. This fix has been tested and the results are satisfactory.

PROBLEM REPORTS

PROBLEM REPORT NUMBER: 6

DATE: 21 JUL 82

ORIGINATOR: Captain Glen T. Logan

PROBLEM NAME: Initialization Via Data Statements

PROGRAM(S) HAVING PROBLEM: VAXTOTAL, ICECAP

PROBLEM DESCRIPTION: The programs use data statements to initialize variables. This is a carryover from the overlay structure of TOTAL. The result is that some variables that are supposed to be reinitialized never get reinitialized because the data statements are read only once during program execution.

PROBLEM SERIOUSNESS: Very

DIFFICULTY OF FIX: Medium

SUGGESTIONS FOR FIX: Find all occurrences and initialize the variables in question via a set statement.

DISPOSITION: All known occurrences have been fixed in VAXTOTAL and ICECAP. However, this PR is left open until all occurrences can be found and fixed.

PROBLEM REPORTS

PROBLEM REPORT NUMBER: 7

DATE: 15 JUN 82

ORIGINATOR: Captain Roslyn J. Taylor

PROBLEM NAME: Use of Lower Case for Terminal Input

PROGRAM(S) HAVING PROBLEM: TOTAL, VAXTOTAL, ICECAP

PROBLEM DESCRIPTION: Programs do not accept lower case input.

PROBLEM SERIOUSNESS: Minor inconvenience

DIFFICULTY OF FIX: Medium

SUGGESTIONS FOR FIX: Make the Pascal portion of ICECAP allow lower, upper, and mixed case for input.

DISPOSITION: Pascal portion of ICECAP allows the use of any case for input, however, until ICECAP is fully implemented, there are portions of the FORTRAN modules in ICECAP that still require upper case for input. This PR should stay open until ICECAP is fully implemented or until the FORTRAN modules are changed to allow the use of any case input.

PROBLEM REPORTS

PROBLEM REPORT NUMBER: 8

DATE: 15 JUN 82

ORIGINATOR: Major Charles J. Gembarowski

PROBLEM NAME: The FORM Command

PROGRAM(S) HAVING PROBLEM: ICECAP

PROBLEM DESCRIPTION: The FORM command is unlike the other commands in ICECAP in that if it is incomplete the user must retype the entire command. The other ICECAP commands allow the user to continue typing an incomplete command.

PROBLEM SERIOUSNESS: Minor inconvenience

DIFFICULTY OF FIX: Medium

SUGGESTIONS FOR FIX: Implement command continuation in the same manner as the other ICECAP commands.

DISPOSITION: Still open because no attempt has been made to implement the command continuation feature.

PROBLEM REPORTS

PROBLEM REPORT NUMBER: 9

DATE: 15 AUG 82

ORIGINATOR: Major Charles J. Gembarowski

PROBLEM NAME: ICECAP Logo

PROGRAM(S) HAVING PROBLEM: ICECAP

PROBLEM DESCRIPTION: ICECAP uses a checkerboard character as a building block to form the large ICECAP logo. The checkboard character does not stand out as much as a solid character.

PROBLEM SERIOUSNESS: Minor

DIFFICULTY OF FIX: *Easy but tedious*

SUGGESTIONS FOR FIX: Replace the checkboard character with the space in inverse video. This requires turning inverse video on and off.

DISPOSITION: Partially implemented. Fix can be patched into the latest version of ICECAP when complete. Fix as time permits.

PROBLEM REPORTS

PROBLEM REPORT NUMBER: 10

DATE: 25 AUG 82

ORIGINATOR: Major Charles J. Gembarowski

PROBLEM NAME: *Error Recovery*

PROGRAM(S) HAVING PROBLEM: ICECAP

PROBLEM DESCRIPTION: Program presently requires that, in case of error, the entire command must be retyped from the beginning.

PROBLEM SERIOUSNESS: Medium inconvenience

DIFFICULTY OF FIX: Unknown, but not expected to be easy

SUGGESTIONS FOR FIX: Unknown, except that some scheme to keep that portion of the command that is correct in the command buffer and continue the command input process from the point of last corrct entry.

DISPOSITION: Still open because there has been *no* attempt to implement an error recovery feature.

C.3 BLANK PROBLEM REPORT FORM

PROBLEM REPORT NUMBER: __________

DATE: _______________

ORIGINATOR: ____

PROBLEM NAME: ____

PROGRAM(S) HAVING PROBLEM: _______________________________

PROBLEM DESCRIPTION: _____________________________________

____

____

____

____

____

____

____

PROBLEM SERIOUSNESS: _____________________________________

DIFFICULTY OF FIX: _______________________________________

SUGGESTIONS FOR FIX: _____________________________________

____

____

____

____

____

DISPOSITION: ____

C.4 SUMMARY

A set of problem reports has been presented. Most of the problems contained herein have already been corrected and the fix has been implemented in ICECAP. This appendix is meant to be the start of a continuous effort to document problems with ICECAP.

APPENDIX D

COMPUTER AIDED DESIGN PACKAGES FOR CONTROL

D.1 INTRODUCTION

This appendix provides a synopsis of representative computer-aided design programs studied during this investigation.

D.2 PROGRAM SYNOPSES

ADAPT
BLZ
BPASS
CADS
CALICO
CESA
DELIGHT-MIMO
DIGIKON
FORTRAC
HONEY-X
I-G SPICE
INTERAC
LPASS
LSAP
MATRIXx
SOFE
SUPER-SCEPTRE
TOTAL
VAXTOTAL

PROGRAM NAME: ADAPT -- Recursive Digital Filters (Kalman Filters)

DESCRIPTION: Reads desired filter parameters (SIGMA, M and Q), generates an initial S (covariance) matrix, T matrix, and W (weight) matrix. The S and W matrices are used to initialize the S and W matrices, respectively. Next, 101 sample points provided by the user are read as input to the Kalman filter. ADAPT tabulates the sample number filter input, filter output, error signal, and Kalman gain.

LANGUAGE: DEC FORTRAN

HOST COMPUTER: PDP-11/20

REFERENCE: [4: 117]

PROGRAM NAME: BLZ -- Bilinear Z-transform

DESCRIPTION: An interactive digital filter design program that calculates digital transfer function coefficients and magnitude function, applies a bilinear transformation with pre-warping to obtain realizable stable digital filters. Consists of a main program and 4 subroutines for pre-warping.

LANGUAGE: DEC FORTRAN

HOST COMPUTER: PDP-11/20

REFERENCE: [4: 81]

PROGRAM NAME: BPASS -- Band Pass Filter Design

DESCRIPTION: Designs maximally flat Butterworth or Chebychev filter with equal ripple in pass band (band pass or band stop). Generates digital filter coefficients for up to six second order sections in cascade (12th order). Must be called as a subroutine from the main program.

LANGUAGE: FORTRAN IV

HOST COMPUTER: PDP-11/20

REFERENCE: [4: 153]

PROGRAM NAME: CADS -- Computer Automated Design of Systems

DESCRIPTION: Simulates and optimizes control systems and circuits. Control system is defined in block diagram form (transfer functions). The transfer functions are reduced to first order differential equations. The unknown or adjustable parameters are set by a minimization routine to acheive the desired reponse. Batch (cards) input.

LANGUAGE: FORTRAN IV

HOST COMPUTER:

REFERENCE: [4: 82]

COMPUTER AIDED DESIGN PACKAGES FOR CONTROL

PROGRAM NAME: CALICO -- Computer Aided Linear Time-Invariant Compensator Optimization Program

DESCRIPTION: For design of compensators to acheive desired response in accordance with selected cost function. Batch (cards) input. Four major parts including subroutines -- 180 K words (210 K with plotting routines)

LANGUAGE: FORTRAN IV

HOST COMPUTER: IBM 360-67

REFERENCE: [42: 32]

PROGRAM NAME: CESA -- Complete Eigenstructure Assignment Program

DESCRIPTION: An interactive program to design a state space control law for multi-input, multi-output systems. Includes regulator, disturbance rejector, and tracker design capabilities.

LANGUAGE: FORTRAN IV

HOST COMPUTER: CDC CYBER

REFERENCES: [36]

PROGRAM NAME: DELIGHT-MIMO (in development)

DESCRIPTION: A highly interactive system for optimization based design of multivariable control systems. Uses color graphics and graphics tablet system interconnections. Employs highly sophisticated semi-infinite optimization algortihms.

LANGUAGE: FORTRAN 77

HOST COMPUTER: VAX 11/780

REFERENCE: [53]

PROGRAM NAME: DIGIKON

DESCRIPTION: Batch and interactive packages for analysis of single-input/single-output control systems. Intended mainly for industrial use. Used for multi-rate digital design. Does root locus, eigenvalue and eigenvector analyses. Packages are designed for both continuous and discrete systems.

LANGUAGE: FORTRAN IV

HOST COMPUTER: IBM 360/370, CDC 6600, Honeywell 66

REFERENCE: [31]

COMPUTER AIDED DESIGN PACKAGES FOR CONTROL

PROGRAM NAME: FORTRAC

DESCRIPTION: For the design of multivariable digital control systems. Can design a discrete control law, can design an observer, and can run a simulation of the system for the resulting controller. Takes the continuous time description of a linear system and synthesizes a control law for discrete-time-optimal regulators, disturbance rejectors, and trackers.

LANGUAGE: FORTRAN

HOST COMPUTER: CDC 6600/CYBER-74

REFERENCE: [5]

PROGRAM NAME: HONEY-X

DESCRIPTION: Interactive package for control system analysis and design intended for research and development applications. Handles multiple-input/multiple-output systems. Does matrix manipulation and Nichols and Nyquist analyses. Finds the time history response of a control system. Handles Kalman filtering and optimal control.

LANGUAGE: FORTRAN 77

HOST COMPUTER: Honeywell 66 (under MULTICS)

REFERENCE: [31]

COMPUTER AIDED DESIGN PACKAGES FOR CONTROL

PROGRAM NAME: I-G SPICE

DESCRIPTION: Interactive graphics version of the SPICE2 program. SPICE2 is a circuit analysis program featuring AC analysis, transient analysis, DC, noise, sensitivity, driving point impedance, Fourier, temperature, distortion, transfer characteristics, and transmission analysis.

LANGUAGE: FORTRAN

HOST COMPUTER: VAX, PRIME, IBM maxi's, CDC maxi's

REFERENCE: AB Associates Announcement

PROGRAM NAME: INTERAC -- An Interactive Software Package for Direct Digital Control Design

DESCRIPTION: Synthesizes a discrete multi-variable feedback gain matrix to control a multi-input, multi-output continuous control system. Three types of design problems are solved: regulator, disturbance rejector, and tracker.

LANGUAGE: FORTRAN IV

HOST COMPUTER: CDC CYBER

REFERENCE: [5]

PROGRAM NAME: LPASS -- Low Pass Filter Design

DESCRIPTION: Designs maximally flat Butterworth or equiripple Chebychev low pass filter. First analog filter is specified, then transformed with bilinear Z-transform to yield the equivalent digital filter. Interactive or batch. Must be called as a subroutine from main program.

LANGUAGE: FORTRAN IV

HOST COMPUTER: PDP-11/20

REFERENCE: [4: 127]

PROGRAM NAME: LSAP -- Linear Systems Analysis Program

DESCRIPTION: An interactive program with graphics capability used for analysis and design of linear control systems. Classical design tools: transfer function manipulation, root locus analysis, frequency response, time response. Analyzes both continuous and sampled data systems. 32K overlay structure.

LANGUAGE: Pascal

HOST COMPUTER: PDP-11/45, RSX-11M operating system

REFERENCE: [32]

PROGRAM NAME: MATRIXx

DESCRIPTION: A data analysis, systems identification, control design and simulation package. It is an interactive software system for computer-aided design and analysis of control systems for dynamic plants. Handles multiple-input/multiple-output systems. Has command interpreter. Solves Riccati equations. Uses state-of-the-art algorithms for linear system analysis, differential equation solution and Fourier trasformation.

LANGUAGE: ANSI FORTRAN 77

HOST COMPUTER: VAX 11/780, planned for IBM 3033, CDC

REFERENCE: [63]

PROGRAM NAME: SOFE -- A Generalized Digital Simulation for Optimal Filter Evaluation

DESCRIPTION: Helps to design and evaluate Kalman filters for integrated systems. SOFE is a Monte Carlo simulation that can be used for system performance analysis once the Kalman filter is designed and verified. A companion post-processor program, SOFEPL, is used for doing ensemble averaging across runs and for making pen plots. Uses batch.

LANGUAGE: '66 ANSI FORTRAN

HOST COMPUTER: CDC CYBER-74

REFERENCE: [49, 50]

PROGRAM NAME: SUPER-SCEPTRE

DESCRIPTION: Analyzes electronic circuits, mechanical systems, logic, transfer functions, and guidance and control systems.

LANGUAGE: FORTRAN

HOST COMPUTER: VAX, PRIME, IBM maxi's, CDC maxi's

REFERENCE: AB Associates announcement

PROGRAM NAME: TOTAL -- Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis

DESCRIPTION: An interactive computer aided design program for continuous and discrete control systems. Classical tools: Block diagram manipulation, root locus analysis, frequency response, time response. Modern Techniques: Matrix manipulation and state-space analysis. Continuous to discrete transformations: impluse invariance, Tustin approximation, first difference approximation. 65K over-lay structure (1 main, 19 primary, 25 secondary) -- total of 600,000 (octal).

LANGUAGE: FORTRAN IV / FORTRAN-77

HOST COMPUTER: CDC CYBER

REFERENCE: [39, 40]

COMPUTER AIDED DESIGN PACKAGES FOR CONTROL

PROGRAM NAME: VAXTOTAL

DESCRIPTION: The implementation of TOTAL (cf.) on the VAX 11/780. Interactive mode of operation at 9600 baud.

LANGUAGE: DEC FORTRAN-77

HOST COMPUTER: VAX-11/780

REFERENCE: [41]

D.3 SUMMARY

Several computer-aided control system design packages have been synopsized. A brief description is given for each along with an indication of the language used and of the type of computers that host the various packages.

APPENDIX E

COMMAND LANGUAGE DEFINITION

E.1 INTRODUCTION

This appendix presents the ICECAP command language definitions in flow chart form. These definitions unambiguously define the ICECAP command language. Definitions are provided in alphabetical order. The standards used to develop the command language diagrams are also provided.

E.2 LIST OF COMMAND LANGUAGE DEFINITIONS

The following is a list of command language definitions that are described in this appendix:

- COPY
- DEFINE
- DISPLAY
- FORM
- HELP
- PRINT
- TURN

E.3 COMMAND LANGUAGE DEFINITION STANDARDS

For the sake of clarity and uniformity the following standards were used in developing the diagrams that portray the command language definitions:

E.3.1 All diagrams are to be read from left to right.

E.3.2 Bracketed terms indicate choices. Only one choice per bracket is allowed.

E.3.3 A lower case command word indicates that the feature has not yet been implemented in the language.

E.3.4 The full spelling of each command word is used in each case. It is understood that the abbreviations as previously described are also valid.

E.3.5 It is understood that the carriage return and the dollar sign are also valid choices at any point in the diagram. The carriage return will cause the system to prompt the user regarding the choices for the next word. The dollar sign will abort the commmand.

E.3.6 It is understood that at least one blank must separate the words in the command string.

E.3.7 The blanks in some of the brackets are there only to give the diagram balance.

$$\text{COPY} \left\{ \begin{array}{l} \text{AMAT} \\ \text{BMAT} \\ \text{CMAT} \\ \text{DMAT} \\ \text{FMAT} \\ \text{GMAT} \\ \text{KMAT} \end{array} \right\} \quad \left\{ \begin{array}{l} \text{AMAT} \\ \text{BMAT} \\ \text{CMAT} \\ \text{DMAT} \\ \text{FMAT} \\ \text{GMAT} \\ \text{KMAT} \end{array} \right.$$

$$\text{COPY} \left\{ \begin{array}{l} \text{CLTF} \\ \text{GTF} \\ \\ \text{HTF} \\ \text{OLTF} \end{array} \right\} \quad \left\{ \begin{array}{l} \text{CLTF} \\ \text{GTF} \\ \\ \text{HTF} \\ \text{OLTF} \end{array} \right.$$

Figure E-1. Command Language Definition for COPY

$$\text{DEFINE} \left\{ \begin{array}{l} \text{AMAT} \\ \text{BMAT} \\ \text{CMAT} \\ \text{DMAT} \\ \text{FMAT} \\ \text{GMAT} \\ \text{KMAT} \\ \text{SETUP} \end{array} \right.$$

$$\text{DEFINE} \left\{ \begin{array}{l} \text{CLTF} \\ \text{GTF} \\ \\ \text{HTF} \\ \text{OLTF} \end{array} \right\} \quad \left\{ \begin{array}{l} \text{FACT} \\ \\ \text{POLY} \end{array} \right\} \quad \left\{ \begin{array}{l} \text{s} \\ \text{w} \\ \\ \text{wprime} \\ \text{z} \end{array} \right.$$

Figure E-2. Command Language Definition for DEFINE

DISPLAY
- AMAT
- BMAT
- CMAT
- DMAT
- FMAT
- GMAT
- KMAT

DISPLAY
- CLTF
- GTF
- HTF
- OLTF
- SPECS

DISPLAY LOCUS
- AUTOSCALE
- MAGNIFY
- SHRINK
- zoom

Figure E-3. Command Language Definition for DISPLAY

FORM
- OLTF
- CLTF USING GTF AND HTF
- CLTF USING OLTF

Figure E-4. Command Language Definition for FORM

$$\text{HELP} \quad \left\{ \begin{array}{l} \text{SYSTEM} \\ \text{INITIAL} \\ \text{teach} \\ \text{COPY} \\ \text{define} \\ \text{print} \\ \text{display} \\ \text{turn} \end{array} \right.$$

Figure E-5. Command Language Definition for HELP

$$\text{PRINT} \quad \left\{ \begin{array}{l} \text{AMAT} \\ \text{BMAT} \\ \text{CMAT} \\ \text{DMAT} \\ \text{FMAT} \\ \text{GMAT} \\ \text{KMAT} \end{array} \right.$$

$$\text{PRINT} \quad \left\{ \begin{array}{l} \text{CLTF} \\ \text{GTF} \\ \text{HTF} \\ \text{OLTF} \\ \text{SPECS} \end{array} \right.$$

$$\text{PRINT} \quad \text{LOCUS} \quad \left\{ \begin{array}{l} \text{AUTOSCALE} \\ \text{MAGNIFY} \\ \\ \text{SHRINK} \\ \text{zoom} \end{array} \right.$$

Figure E-6. Command Language Definition for PRINT

$$\text{TURN} \quad \left\{ \begin{array}{l} \text{ANSWER} \\ \text{CANCEL} \\ \text{CAPTION} \\ \text{CLOSED} \\ \text{DEBUG} \\ \text{DECIBELS} \\ \text{ECHO} \\ \text{GRID} \\ \text{HERTZ} \\ \text{MAINMENU} \\ \text{MULT} \\ \text{PLOT} \\ \text{SCALE} \\ \text{TITLE} \end{array} \right\} \quad \left\{ \begin{array}{l} \text{ON} \\ \\ \text{OFF} \end{array} \right.$$

Figure E-7. Command Language Definition for TURN

E.4 SUMMARY

This appendix has provided an unambiguous definition of the ICECAP command language in diagram form. The standards which were used in developing the diagrams have teen provided to help the reader understand the diagrams and to serve as a guideline for others who will be extending the language to add more commands.

APPENDIX F

FORTRAN MODULE DESCRIPTIONS

F.1 INTRODUCTION

This appendix gives descriptions of new FORTRAN modules used in ICECAP and descriptions of the VAXTOTAL FORTRAN modules that were revised so they could be used in ICECAP. A complete source listing of these modules appears in alphabetical order in Appendix G.

F.2 DESCRIPTION OF NEW FORTRAN MODULES

The following FORTRAN modules were developed as a result of this thesis effort. These modules were coded in FORTRAN as they are more closely related to the FORTRAN portion of ICECAP than they are to the Pascal portion. TOTICE is the interface module between Pascal portion of ICECAP and the FORTRAN modules. Since most of its code was derived from the mainline program of VAXTOTAL which is written in FORTRAN, it made sense to leave the code in FORTRAN.

FORTRAN MODULE DESCRIPTIONS

- FINDBORD - This module is used to establish the borders for the plot of the root locus. The location of the borders is calculated based on the location of the poles and zeroes.

- MAGNIFY - This module is used to double the size of the root locus as it appears on the plot. This is done by dividing the location of each boundary by two.

- SHRINK - This module is used to shrink the size of the root locus as it appears on the plot by a factor of two. This is done by multiplying the location of each boundary by two.

- TOTICE - This module is the main interface between the new ICECAP modules and the old VAXTOTAL modules. ICECAP takes commands that have been formulated by the user and translates them to option numbers and commands that VAXTOTAL normally processes. TOTICE passes these option numbers and commands to the VAXTOTAL modules for action and then returns control back to ICECAP.

- TOTINI - This module is the initialization module for the VAXTOTAL modules that are used in ICECAP.

F.3 DESCRIPTION OF REVISED FORTRAN MODULES

The following is a list of the FORTRAN Modules that have undergone major revision. A summary of the changes made to each module is included. The modules are in alphabetical order.

- ADAPT - FORTRAN output statements were reformatted so as to be compatible with the appearance of the Pascal generated output messages. Cursor controls were added so that the cursor appears right after the prompt for user data entry rather than on the line following the prompt.

- ANG1 - PI was made double precision.

- BANG - PI was made double precision.

FORTRAN MODULE DESCRIPTIONS

- o BLOCKER - FORTRAN output statements were reformatted so as to be compatible with the appearance of the Pascal generated output messages.

- o BOX - PI was made double precision.

- o COPYIER - FORTRAN output statements were reformatted so as to be compatible with the appearance of the Pascal generated output messages. Page slewing was added for when the program prints to the file ANSWER.DAT.

- o DECODER - FORTRAN output statements were reformatted so as to be compatible with the appearance of the Pascal generated output messages. Cursor controls were added so that the cursor appears right after the prompt for user data entry rather than on the line following the prompt. References to filenames were changed to be compatible with names used in VAX/VMS. Messages refering to ANSWER flag being turned on and off were suppressed.

- o DMULR - Code was stripped out and replaced with a call to ROOT which in turn calls library routine ZRPOLY which uses the Jenkins-Traub method of finding roots of polynomials.

- o FACTO - Unnecessary declarations were commented out of the code.

- o FRACTOR - Code was stripped out and replaced with a call to FACTO which in turn calls ROOT which in turn calls library routine ZRPOLY which uses the Jenkins-Traub method of finding roots of polynomials.

- o GANG1 - PI was made double precision.

- o PARTL - FORTRAN output statements were reformatted so as to be compatible with the appearance of the Pascal generated output messages. Cursor controls were added so that the cursor appears right after the prompt for user data entry rather than on the line following the prompt.

- o POLY - FORTRAN output statements were reformatted so as to be compatible with the appearance of the Pascal generated output messages. Cursor controls were added so that the cursor appears right after the prompt for user data entry rather than on the line following the prompt.

- READER - The parameter ICELINE was added to calling statement. A statement to set the FORTRAN common variable LINE equal to ICELINE was added.

- ROOT - Code was stripped out and replaced with library routine ZRPOLY which uses the Jenkins-Traub method of finding roots of polynomials.

- ROOT10 - PI was made double precision. A condition on when to slew the page was added. The display of AA, BB, CC, DD was supressed at the terminal. Code to prevent VAXTOTAL Option 49 from erroneously resetting VAXTOTAL variables DEL and DELPR was added. This erroneous reset had caused problems in finding the Root Locus.

- ROOT11 - FORTRAN output statements were reformatted so as to be compatible with the appearance of the Pascal generated output messages.

- ROOT12 - FORTRAN output statements were reformatted so as to be compatible with the appearance of the Pascal generated output messages.

- ROOT2 - This module was deleted since it is no longer needed because ZRPOLY is being used instead to find the roots of polynomials.

- SEEK - PI was made double precision.

- SMULR - Code was stripped out and replaced with a call to ROOT which in turn calls library routine ZRPOLY which uses the Jenkins-Traub method of finding roots of polynomials.

- SPECS - FORTRAN output statements were reformatted so as to be compatible with the appearance of the Pascal generated output messages.

- SWAP - Cursor controls were added so that the cursor appears right after the prompt for user data entry rather than on the line following the prompt.

- SWAPER - FORTRAN output statements were reformatted so as to be compatible with the appearance of the Pascal generated output messages. Cursor controls were added so that the cursor appears right after the prompt for user data entry rather than on the line following the prompt.

- o TTYPLOT - Cursor controls were added. Automatic scaling for Root Locus plots was added. AA, BB, CC, DD were changed to Right, Top, Left, Bottom respectively.
- o UPDATE - Code was added to force control to return to ICECAP rather than to do a FORTRAN stop.

F.4 SUMMARY

This appendix has provided descriptions of the new FORTRAN modules being used in ICECAP and descriptions of the VAXTOTAL FORTRAN modules that were revised so they could be used in ICECAP. The next appendix gives a complete source listing of these modules in alphabetical order.

VITA

Major Charles J. Gembarowski was born on 20 September 1944 in Brattleboro, Vermont. He graduated from Saint Michael's High School in 1962. He received a Bachelor of Arts Degree in Philosophy from Saint Mary's Seminary and University in Baltimore, Maryland in 1966. He received a Bachelor of Science Degree in Electrical Engineering and a Bachelor of Arts Degree in Mathematics from Arizona State University in 1974. Major Gembarowski was the Engineer for the onboard central computer for the E-3A Airborne Warning and Control System (AWACS) while assigned to Hanscom Air Force Base, Massachusetts. His most recent assignment before entering the School of Engineering of the Air Force Institute of Technology was as a Program Manager for the Modular Automatic Test Equipment (MATE) Program at Wright-Patterson Air Force Base, Ohio. Major Gembarowski is a member of Eta Kappa Nu and Tau Beta Pi. He is married to Ruthmary and has two children, Charles and Christopher.

Permanent Address: Brattleboro Road
North Hinsdale,
New Hampshire 03451

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/82D-34	2. GOVT ACCESSION NO. AD-A124707	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DEVELOPMENT OF AN INTERACTIVE CONTROL ENGINEERING COMPUTER ANALYSIS PACKAGE (ICECAP) FOR DISCRETE AND CONTINUOUS SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Charles J. Gembarowski		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Aeronautical Systems Division Flight Control and Stability Branch (ASD/ENFTC) Wright-Patterson AFB, Ohio 45433		12. REPORT DATE December 1982
		13. NUMBER OF PAGES 171
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release: IAW AFR 190-17.
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

4 JAN 1983

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Command Language	Control Systems
Computer Aided Design	Discrete Time Systems
Computer Analysis Package	Human Interface
Continuous Time Systems	Interactive
Control Engineering	VAX 11/780

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See reverse

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This thesis reports on an effort to design and implement a modern interactive computer-aided design and analysis package for control systems. This package applies to discrete and continuous time systems. The thesis effort continues the effort begun by Captain Glen T. Logan who used a control engineering design and analysis computer program called TOTAL as his starting point.

This thesis project uses top-down structured analysis and programming techniques to define the new program called ICECAP (Interactive Control Engineering Computer Analysis Package). A user-oriented command language forms the basic structure of ICECAP. On-line assistance is provided to the user. The program makes use of CRT (Cathode Ray Tube) terminals with a limited graphics capability to improve the user environment.

The program structure allows features to be added in a modular fashion so that others can continue the effort. Emphasis was placed on implementing the continuous time functions first.

LME

8